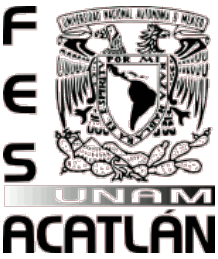


UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

"TUTORIAL DE ADMINISTRACIÓN DEL SISTEMA OPERATIVO UNIX SOLARIS".

T E S I S

QUE PARA OBTENER EL TITULO DE:

Licenciatura en Matemáticas Aplicadas y Computación

PRESENTA:

LILIANA CASTORENA SÁNCHEZ.

Asesor:
Lic. Guadalupe del Carmen Rodríguez Moreno.

Junio 2007.

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Dedicatorias

A mis padres:
Por su gran apoyo
cariño y confianza
por su esfuerzo y tiempo que
me dieron para logrará concluir
mis estudios.

A mi hermana Gris:
Por ser un ejemplo a seguir.
y por su gran cariño

A mi hermana Renata:
Por su tiempo que dedicado a mi
para que lograra finalizar
con este trabajo.

A mi cuñado Antonio:
Por apoyarme en todo momento.

A Carlos Bonifacio:
Por la gran confianza que tuviste
en mi y tu gran apoyo para que
concluyera mi carrera, mil gracias.

A mi abuelita a mi familia y a Dios:
Por acompañarme en todo
momento.

A mis hijas Meggan y Melanie:
Por que este trabajo sea un ejemplo y motivo
para que concluyan todo lo que comiencen en su vida.

Por que nunca es tarde para lograr las metas.
(Fernando, Daniela, Merino, Odiseo y amigos)

ÍNDICE

INTRODUCCIÓN

El propósito de este tutorial es dar a conocer los conceptos y terminologías básicas del Sistema Operativo UNIX Solaris para que un usuario en general obtenga: el conocimiento necesario y sea capaz de administrar de una manera rápida y sencilla el sistema; a través de la comprensión de la estructura básica y capacidades del mismo, además de adquirir la experiencia andando por los capítulos, donde se describen los problemas, la optimización, la automatización, y los mantenimientos, así como sus propuestas en un ambiente de trabajo real.

Este trabajo no únicamente es una recopilación de datos. Es un tutorial, es una forma sencilla y guiada para el rápido aprendizaje. La estructura del capitulado se realizo de tal forma que lleva paso a paso al usuario sin confundirlo. El lenguaje utilizado es sencillo y comprensible para cualquiera, y algunos términos se definieron en el glosario "ANEXO A". Adicionalmente cada apartado lleva una sintaxis y unos ejemplos de cada comando utilizado en el sistema operativo UNIX Solaris, para que el usuario avance ya sea con o sin una sesión a tiempo real para empezar a administrar su sistema.

Este tutorial no esta acompañado por algún software ya que la idea principal es tenerlo en cualquier momento y ocasión como una guía rápida, a demás en la actualidad no es tan publico el acceso a este tipo de tutoriales

El lenguaje utilizado es sencillo y comprensible para cualquier usuario, y algunos términos se definieron en el ANEXO A.

En este trabajo se utilizaron recuadros para visualizar la sintaxis de comandos, no es la salida de pantalla del sistema operativo UNIX Solaris. También se manejan tablas que hacen referencia a algunas de las opciones existentes propias de algunos comandos.

Cada apartado lleva sintaxis y ejemplos de cada comando utilizado en el sistema UNIX para que el usuario avance ya sea con o sin una sesión para empezar a administrar su sistema.

El usuario debe tener el conocimiento básico de la administración del sistema operativo UNIX con este tutorial, el usuario podra adquirir conocimientos avanzados en comandos y estructura del sistema operativo UNIX Solaris.

CAPÍTULO 1. Orientación general al Sistema Operativo UNIX

1.1 Introducción al Sistema Operativo UNIX

El Sistema UNIX Solaris proporciona un Sistema Operativo de tiempo compartido que controla las actividades y recursos de la computadora, y una interfaz *interactiva,* operativa y flexible. Fue diseñado para ejecutar múltiples procesos concurrentes y soporta múltiples usuarios para facilitar el compartimiento de datos entre los miembros de un grupo. El ambiente operativo fue diseñado con una arquitectura modular en todos los niveles.

Desde su creación, el Sistema Operativo UNIX se desarrolló con un enfoque de portabilidad, sin restricciones a un procesador o plataforma y debido a que fue escrito en un lenguaje de alto nivel, es fácil de modificar.

El UNIX es un Sistema Operativo que tiene las siguientes ventajas:

***Multitareas*:** En el Sistema UNIX se pueden realizar varias tareas al mismo tiempo. Desde una simple terminal, un usuario puede ejecutar simultáneamente varios programas ya que UNIX tiene la capacidad de compartir el tiempo de CPU entre múltiples procesos.

***Multiusuario*:** Múltiples usuarios pueden trabajar a la vez.

***Multisesión*:** Un usuario puede abrir varias sesiones de una o más aplicaciones.

La filosofía UNIX se basa en la idea de que un sistema informático potente y complejo debe ser *simple, general y extensible*; por ello proporciona importantes beneficios tanto para los usuarios como para los que desarrollan programas.

Se definirán los términos más comunes para el Sistema Operativo UNIX Solaris en el ANEXO A.

1.2 Entrando al Sistema Operativo UNIX

El acceso al Sistema Operativo UNIX se realiza desde una terminal ya sea local o remota, si el sistema consta de un periférico de despliegue gráfico el acceso se llevará a cabo en forma local, aunque se puede utilizar una PC empleando un emulador que opera bajo el sistema operativo Windows.

El administrador del sistema proporcionará un nombre de usuario para el inicio de sesión “user-name” llamado también login ID y le asignará una contraseña “password”.

1.2.1 Restricciones para user-name y password

El user-name ó login y el password cuentan con un cierto número de requisitos para su buen funcionamiento, a continuación veremos algunos de estos requisitos:

User-name

- Debe tener más de seis caracteres como mínimo. El user-name es sensible a el uso mayúsculas y minúsculas, por lo que el usuario “user1” es diferente al usuario “USER1”
- Puede contener cualquier combinación de letras, números y caracteres especiales.
- El user-name no puede tener ningún símbolo ni espacio y debe ser único para cada usuario.

Password (Contraseña)

- Por lo general el password debe tener al menos seis caracteres formados por lo menos dos caracteres alfabéticos y al menos un carácter numérico. Se puede utilizar letras minúsculas y mayúsculas.
- El password no puede contener espacios.
- El nombre de usuario, con las letras invertidas o desplazadas, no pueden utilizarse como password. Por ejemplo si el user-name es paco23, el password 32ocap, no serán aceptadas.
- En un cambio de password, los caracteres en mayúscula y minúscula no se consideran diferentes. Por ejemplo de paco23 a PACO23, el sistema no le permitirá cambiar el password.
- Un nuevo password debe ser diferente de la anterior en al menos tres caracteres.

1.2.2 Formato de línea de comando

Durante el proceso de entrada al Sistema UNIX se inicia un shell que es el responsable de enviar el prompt, e interpretar los comandos que se suministren.

Los comandos se pueden definir como programas que acompañan al sistema operativo sin formar parte de él, estos están diseñados para trabajar con los comandos propios del kernel (o núcleo) de UNIX.

Después del proceso de entrada enviará el prompt del shell, comúnmente es el signo de pesos $ para el caso de un usuario sin privilegios y un # en el caso del superuser, en este momento se podrá ingresar un comando.

Un espacio en blanco se usa para delimitar comandos, opciones y argumentos. Cada comando será terminado con un Return (Enter), esto trasmitirá el comando al sistema para su ejecución, si el comando envía información o resultados a pantalla se desplegarán en la línea siguiente sin prompt como se muestra en la figura 1.2.2:

FORMATO DE LÍNEA DE COMANDO

Sintaxis:

$ comando [-*opciones] [argumentos*]

Ejemplos:

$ date

Mon Jul 01 11:07:43 EDT 2006

Figura 1.2.2 Formato de línea de comando.

1.2.3 Saliendo del sistema

Para finalizar una sesión de terminal "*logoff* ", en el prompt $ teclee el comando: *exit*. Si es necesario se repetirá esta tarea hasta que se cierre la sesión como lo muestra la figura 1.2.3.

FINALIZAR UNA SESIÓN

Sintaxis:

$ exit

Figura 1.2.3 Finalizar una sesión.

1.3 Mantenimiento de usuarios

Para realizar las tareas de administrador del sistema será necesario trabajar bajo el usuario "*root*" llamado también "superusuario" ya que este usuario cuenta con todas las capacidades, privilegios y accesos sobre todos los archivos del sistema (por ejemplo: puede borrar, modificar, acceder, etc. a cualquier archivo sin importar los permisos y dueño al que pertenezcan dichos archivos) a demás el sistema operativo esta configurado bajo el usuario root y todos los archivos pertenecen a este usuario, por ello es que se deberá tener mucha cautela en el uso de el usuario "root". El administrador del sistema tiene la responsabilidad de definir el entorno del usuario, añadir usuarios, determinar el tiempo de validez de los passwords, así como eliminar usuarios entre muchas tareas más.

Un user-name (nombre de usuario) contiene información necesaria para entrar y trabajar en el sistema. Esta información consiste de cuatro componentes principales:

user-name	(nombre de usuario)
pasword	(contraseña)
home-directory	(directorio propio del usuario)
shell	(interprete de comandos)

1.3.1 Archivos de seguridad e integridad del usuario

El Sistema UNIX se diseñó de modo que los usuarios puedan entrar fácilmente a sus recursos y compartir información con otros usuarios, por ello la seguridad es relativa. Proporcionar seguridad que sea altamente resistente en el sistema requiere de procedimientos y técnicas especiales, por ello, este tutorial no se adentrará en el tema, simplemente se analizarán algunos archivos básicos.

El Sistema UNIX guarda información acerca de los usuarios en los archivos, /etc/passwd, /etc/shadow y /etc/group (que se encuentran bajo la ruta /etc). Estos archivos son usados por el sistema para validar los usuarios y preparar el entorno de trabajo inicial.

1.3.1.1 El archivo /etc/passwd

El archivo passwd se encuentra en la ruta /etc/passwd y es una parte integral de la seguridad del Sistema UNIX. Existe una línea en /etc/passwd por cada usuario, cada una de estas líneas contiene secuencias de campos, separados por dos puntos. En la figura 1.3.1.1 se muestra la definición de los campos y un ejemplo del contenido el archivo passwd:

ARCHIVO /etc/passwd

user-name:x:UID:GID:coment:home-dir:login-shell

Ejemplo del contenido del archivo passwd

lcs:x:1911:21:lilCastor:/home/lil:/bin/rsh

Figura 1.3.1.1 Archivo /etc/password.

El archivo passwd guarda cierta información del usuario como se mostró anteriormente, la tabla 1.3.1.1 describe cada campo:

Archivo passwd	
user-name	Nombre del usuario.
X	Representa la encriptación del password del usuario.
UID	Número de identificación del usuario.
GID	Identificador de grupo al que pertenece el usuario.
Coments	Contiene información del usuario.
home-dir	Directorio propio del usuario.
login-shell	Determina el interprete de comandos con el que el usuario trabajará.

Tabla 1.3.1.1 Archivo /etc/passwd

1.3.1.2 El archivo /etc/shadow

El archivo shadow se encuentra en la ruta /etc/shadow y contiene información acerca de los password de los usuarios y datos referentes al tiempo de validez (expiración) del password. Existe una línea en /etc/shadow por cada usuario, cada una de estas líneas contiene secuencias de campos, separados por dos puntos como se muestra en la figura 1.3.1.2

ARCHIVO /etc/shadow

user-name:password:lastchg:min:max:warn:inactive:expire

Ejemplo:
lcs:ubQwoeZXMDdjuv:8532:7:100:5:20:7400:

Figura 1.3.1.2 Archivo /etc/shadow

El archivo shadow guarda cierta información del usuario como se muestra en la tabla 1.3.1.2:

Archivo shadow	
user-name	Nombre del usuario.
password	Contraseña encryptada: - NP no tiene pasword. *LK* cuenta inaccesible.
lastchg	Indica el número de días entre el 1° de Enero de 1970 y el día en que la contraseña fue modificada la ultima vez.
min	El mínimo número de días entre cambios de la contraseña.
max	El máximo número de días que una contraseña es válida.
warn	El número de días antes de la expiración de una contraseña para enviar el aviso al usuario.
inactiv	Indica los días de inactividad permitidos para el usuario.
expire	Contiene la fecha (año, mes y día) absoluta a partir de la cual la cuenta del usuario ya no puede ser utilizada.

Tabla 1.3.1.2 Archivo /etc/shadow

El archivo /etc/shadow sólo puede ser leído por el superusuario ya que contiene las contraseñas encryptadas y es una forma de reducir la vulnerabilidad en cuanto la seguridad del sistema, esto es a nivel usuario, la encriptación evita descubrir los password y con ello se tenga mayor seguridad de que no se filtre por este medio algún usuario extraño al sistema.

1.3.1.3 El archivo /etc/group

Todos los usuarios son miembros de uno o más grupos. El grupo puede ser llamado por su nombre o su ID Group (GID) este último puede ser proporcionado por el administrador al ser creado, si no el sistema asignará este número secuencialmente; por ejemplo: el grupo llamado admon1 es llamado grupo 100. El archivo group se encuentra en la ruta /etc/group y contiene todos los grupos existentes en el sistema, especificaciones y grupos adicionales al cual el usuario pertenece como se muestra en la figura 1.3.1.3.

ARCHIVO /etc/group

group-name:password:GID:user-list

Ejemplo del contenido del archivo group:
root::0:root
other::1:
bin::2:root:bin:daemon
sys::3: root:bin:sys:daemon
admon1::100:estu1:estu2
admon2::101:estu1:estu3:estu4

Figura 1.3.1.3 Archivo /etc/group

El archivo group contiene los siguientes cuatro campos separados por dos puntos ‘:’ y estos se definen en la siguiente tabla:

Archivo group	
Group-name	Nombre del grupo, puede ser hasta de 8 caracteres.
Password	Este campo es inusual y es para el password del grupo.
GID	Es el número de identificación del grupo.
User-list	Lista de usuarios quienes pertenecen al grupo, se colocan separados por comas y hace referencia al grupo secundario.

Tabla 1.3.1.3 Archivo /etc/group

Es recomendable que se generen nuevos grupos con su GID a partir del 100, para reservar los números propios del sistema esto con el fin solo de tener una buena administración del sistema.

Ejemplo group:

- Para añadir un grupo de nombre admon1 al sistema y con GID=100, teclee lo siguiente.

$ groupadd –g 100 admon1

- El comando para eliminar un grupo de nombre admon1.

```
$ groupdel   admon1
```

- Se pueden modificar el nombre del grupo admon1 por el nombre prueba1.

```
$ groupmod  -n  prueba1  admon1
```

Un usuario tiene un grupo primario que es el principal y que se muestra en los archivos password y shadow pero también puede pertenecer a otros grupos, a estos se les llaman secundarios, puede pertenecer hasta 15 grupos secundarios, teniendo un total de 16 grupos. Con estos grupos añadidos al usuario, se tiene la ventaja de poder tener ciertos privilegios en archivos pertenencianetes a estos grupos, como un ejemplo: escribir o modificar el contenido de estos archivos. Estos grupos secundarios pueden ser añadidos en la lista del /etc/group como se vió en la Tabla 1.3.1.3.

Los grupos a los que pertenece un usuario los podemos desplegar con el comando *groups*. Este mostrará los grupos del usuario en el momento de estar conectado, por ejemplo:

Ejemplo groups:

- El siguiente ejemplo se desplegaran los grupos del usuario ‘estu1’ ya que se asumirá que se entró a la sesión con este usuario. Se puede ver que el usuario ‘estu1’ tiene un grupo primario admon1 y un grupo secundario llamado admon2.

```
$groups
admon1     admon2
```

1.3.1.4 Perfil de usuario (profile)

El perfil del usuario es el que delimita el entorno de trabajo en el Sistema UNIX, este entorno contiene el tipo de terminal (existen una gran variedad de terminales, por ejemplo vt100, suncms, vt200, etc.), visualizar la fecha actual al entrar, comprobar si tiene correos, etc. y es suministrado al usuario automáticamente cuando entra a la sesión. Los archivos encargados de esta tarea son: /etc/profile, $HOME/.profile, $HOME/.cshrc, $HOME/.kshrc y $HOME/.login.

1.3.1.4.1 El directorio /etc/skel

El directorio skel contiene los esquemas (templates) previamente definidos dentro del sistema UNIX Solaris de los archivos profile, cshrc y login, de los cuales se basa el sistema para la creación de nuevos usuarios. Estos archivos tienen definidos los siguientes nombres: local.profile, local.cshrc y local.login.

Podemos encontrar el tipo de entorno del usuario, según sea necesario, en el directorio /etc/skel:

Tipos de shell	
/etc/skel/local.cshrc	El shell C
/etc/skel/local.login	El shell C
/etc/skel/local.profile	El shell Bourne o korn

Tabla 13.1.4.1 Tipo de entorno del usuario

ENTORNO DEL USUARIO.

Copiar el archivo local.profile al directorio propio del usuario

con el siguiente formato:

```
$ copy  /etc/skel/local.profile    /home/estu1/.profile
```

Figura 1.3.1.4.1 Entorno del usuario

1.3.1.4.2 Definición de variables de ambiente

El tipo de ambiente para un usuario se pueden establecer de forma global o particular, a continuación se detallan cada uno de estos:

- **El archivo /etc/profile**

  Este archivo define las variables de ambiente de forma global, o sea, para todos los usuarios dentro del sistema, que por definición utilizan los interpretes de comando csh, sh y ksh, que se detallaran más adelante.

- **El archivo $HOME/.profile**

  Este archivo define las variables de ambiente de forma particular para cada usuario del sistema que utilicen los interprete de comandos sh, ksh y bash. Estas se definen por cada usuario en particular y se colocan en su directorio personal $HOME.

1.3.2 Administración de usuarios

Se utilizan los comandos propios para crear o borrar un usuario del Sistema UNIX, se puede realizar con comandos en línea o con el Admintool (Ambiente Gráfico del Sistema UNIX) para versiones Solaris 2.8 y anteriores y para la versiones 9 y posteriores existe el Solaris Management Console. Este tutorial se enfocará a realizar estas tareas por línea de comandos.

1.3.2.1 Comando userdd

La figura 1.3.2.1 explica paso a paso como se genera un usuario con sus campos definidos para el usuario 'estu1' y en la tabla siguiente se muestran algunos argumentos para el comando useradd:

Crear un usuario:

Comando useradd

$ useradd –u UID –g GID –c "comentario" –d <home direcotry> -m \
-s shell user-name

Ejemplo:
$ useradd –u 115 –g 100 –c "Usuario estu1" –d /home/estu1 –m \
-s /bin/sh estu1

Figura 1.3.2.1 Creación de un usuario

Argumentos del comando useradd**:**

Useradd		
ARGUMENTO	**DEFINICION**	**COMENTARIO**
-u	UID	Número de identificación del usuario. Se utiliza el número siguiente del último usuario ingresado. Esto se puede verificar en el archivo /etc/passwd.
-o	UID no único	Se utiliza junto con la opción –u para asignar un uid ya existente a varios usuarios estos con nombres y directorios diferentes pero con el propósito de que tengan los mismos permisos de acceso a archivos.
-g	GID (Primario)	Identificador de grupo primario al que pertenecerá el usuario. Escoger un grupo existente /etc/group o crear un grupo nuevo para este usuario.
-G	GID (Secundario)	Identificador de grupo secundario al que pertenecerá el usuario.
-c	Comentario	Contiene algún tipo de dato del usuario.
-d	Directorio	Indica el nombre de home directory del usuario.
-m	Creación dir	Crea el directorio propio del usuario en caso de no existir.
-s	Shell	Determina con que interprete de comando trabajará el usuario.
-k	Skel_dir	Define el directorio que contiene información esquemática (.profile) Copiar en el directorio propio del usuario el profile de /etc/skel.
-e	Expira	Define la fecha en que un usuario expirará, después de esta fecha no podrá entrar al sistema.
-f	Inactivo	Establece el número de días que un usuario puede estar inactivo.

Tabla 1.3.2.1 Argumentos del comando useradd

1.3.2.2 Comando usermod

El comando usermod permite modificar las características de un usuario ya creado, como por ejemplo modificar el UID, GIU, home-directory o simplemente cambiar el nombre de un usuario.

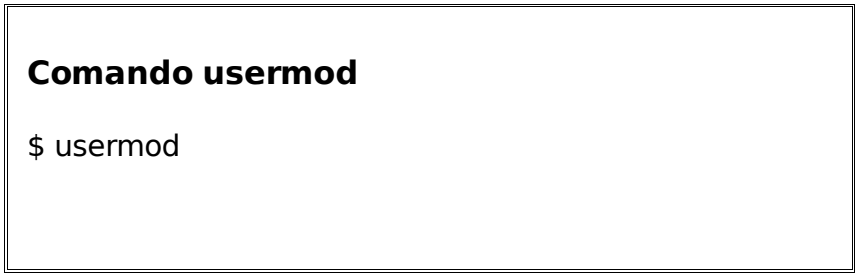


Figura 1.3.2.2 Comando usermod

También puede limitar o bloquear al usuario cambiando el shell del usuario, el comando *usermod* puede ser usado para esta tarea, este modificará la definición del usuario en el sistema y cambiará el shell del usuario al shell restringido, esto es limitar el acceso del usuario a ciertas órdenes y archivos, por ejemplo

Ejemplo usermod:

- En el siguiente ejemplo se restringe a un usuario, el permiso para acceder a un sistema remoto (rsh)

$ usermod –s /user/bin/rsh user-name

1.3.2.3 Comando userdel

Si no se requiere que un usuario esté más en el sistema, puede usarse el comando *userdel* para eliminar definitivamente a un usuario del sisitema, como muestra la figura 1.3.2.3

Borrar un usuario: Comando userdel

$ userdel user-name

Borra el usuario del sistema.

$ userdel –r user-name

Borra el usuario del sistema y borra el directorio propio del usuario.

Figura 1.3.2.3 Borrando un usuario

1.3.2.4 Comando passwd

Ya creado el usuario es indispensable, por seguridad del sistema, ingresar un password. Un nuevo usuario estará bloqueado hasta que se le añada una contraseña. *Ver capítulo 1.2.1 Restricciones para user-name* y *password***.** La figura 1.3.2.4 muestra como ingresar el password:

COMANDO paswd:

```
$paswd  user-name
Enter password for login:
New password:
$
```

Figura 1.3.2.4 Contraseña del usuario

Nota: Si se desea dejar sin password al nuevo usuario, al ingresar el comando passwd, sólo se da un "*enter*" para dejarlo sin contraseña.

1.3.2.4.1 Tiempo de validez del password (contraseña)

Para definir el tiempo de validez de la contraseña, primero hay que conocer la definición de los argumentos del comando passwd ya que con ellos podremos establecer estos tiempos, como lo muestra la tabla siguiente:

Argumentos del comando passwd:

Passwd		
ARGUMENTO	**DEFINICIÓN**	**COMENTARIO**
-d	#paswd -d user-name	Borra la entrada del password del user-name.
-f	#paswd -f user-name	Requiere que se cambie el password la primera vez que el user-name se conecte al sistema.
-x	#paswd -x user-name	Número de días máximo que un password va a funcionar.
-n	#paswd -n user-name	Número de días mínimo que un password puede funcionar.
-l	#paswd -l user-name	Bloquea el acceso al sistema del usuario.

Tabla 1.3.2.4.1 Argumentos del comando passwd.

El archivo /etc/default/login contiene la línea PASSREQ=YES, esto obligara que todos los usuarios introduzcan un passwod para poder entrar a su sesión cada vez que se requiera ingresar al sistema.

Para establecer el tiempo de validez de un password a nivel general, es decir para todos los usuarios en el sistema, hay variables dentro del archivo /etc/default/passwd que contienen los valores para realizar esta tarea, estos son:

MAXWEEKS = 4	Máximo número de semanas que una contraseña puede estar en uso.
MINWEEKS = 1	Mínimo número de semanas permitidas para cambiar una contraseña.
MARNWEEKS = 1	Número de semanas para enviar aviso de que la contraseña debe ser cambiada.

El Sistema UNIX define estos campos como apagados, así que se pueden modificar según nuestras necesidades.

1.3.2.5 Cambiando de un usuario a otro

El comando ***su*** permite cambiar interactivamente su identificador de usuario (user-name o login) a otro usuario en sistema. Esto permite que se inicie una subsesión con el nuevo usuario. Este cambio solicitará el ingreso de la contraseña propia del usuario al cual se pretenda cambiar, como lo muestra la figura 1.3.2.5.

Cambiando de usuario:

```
$ su - [user-name]
Password:
```

Figura 1.3.2.5 Comando su

Esto es: Se inicia la sesión con el usuario ‘estu1’ y en esta misma sesión sin salir del usuario ´stud1´podemos iniciar una nueva con el usuario ‘estu2’, como en el siguiente ejemplo:

Ejemplo su:

- Con el comando id se despliega la información del user-id actual, esto es que indica que se inicio la sesión con el usuario ‘estu1’. Posteriormente se procede a cambiar al usuario ‘estu2’:

```
$id
uid=303 (estu1) ,  gid=300  (admon3)

$su – estu2
Password:
```

- Comprobamos que ya se inicio la sub-sesión con el usuario ‘estu2’

```
$id
uid=205 (estu2) ,  gid=300  (admon3)
```

- Para regresar salir de la subsesión, se utiliza el comando *exit*, éste cerrará la sesión ´estu2´

CAPÍTULO 2. Sistemas de Archivos

2.1 ¿Qué es un sistema de archivos?

Este capítulo proporcionará a los nuevos usuarios una introducción al sistema de archivos UNIX. Se definirán las características del sistema de archivos, se aprenderá a manipular archivos y directorios y a visualizar el contenido de un archivo.

El archivo es la unidad básica del Sistema UNIX. Un archivo es un contenedor de datos. Conceptualmente un archivo es similar a un documento de papel. Técnicamente un archivo es una secuencia de bytes que se almacena en un dispositivo como muestra la siguiente figura 2.1:

Figura 2.1 ¿Qué es un archivo?

Todo en el Sistema UNIX es un archivo, a continuación definiremos los tipos de archivos en el sistema:

Archivo regulares: Texto, mensajes de correo, datos, código fuente de programas. Programas ejecutables como *ksh, who, date, man, ls, etc*.

Directorio: Contiene la lista de identificación de un conjunto de archivos que este contiene.

Liga: Es una referencia que apunta a otro archivo.

Dispositivo: Archivos especiales que proporcionan la interfaz a los dispositivos de hardware tales como discos, terminales, impresoras, memorias, etc.

2.2 Características de un archivo

Algunas de las cararcterísticas asociadas a un archivo son:nombre del archivo, tipo de archivo, permisos de acceso, etc, las cuales se verán en este capítulo.

Un archivo tiene varias características asociadas con él, como se definió en el capitulo anterior. Estas pueden ser desplegadas usando el comando *ls –l* como se muestra en la siguiente figura:

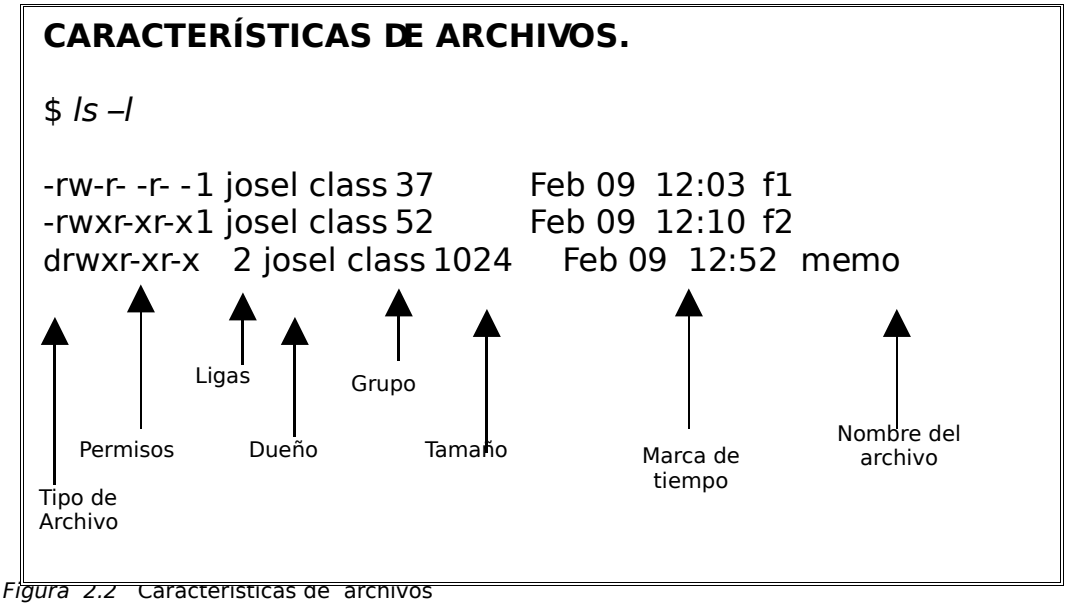


Figura 2.2 Características de archivos

En la siguiente tabla se definirán algunas de estas características:

Características de Archivos.	
Tipo	Archivo regular o archivo especial.
Permisos	Definición de acceso para el archivo.
Ligas	Número de nombres de archivos asociados con una sola colección de datos.

Dueño	Identificación del usuario dueño del archivo.
Grupo	Identificación del grupo para el acceso del archivo.
Tamaño	Número de bytes que contiene el archivo.
Marca de Tiempo	Fecha de última modificación del archivo.
Nombre	Nombre del archivo.

Tabla 2.2 Características de archivos

2.2.1 Nombre de un archivo

El nombre de archivo puede ser casi cualquier secuencia de caracteres. El Sistema UNIX dispone de pocas restricciones sobre cómo nombrar archivos, estas se muestran en la siguiente figura 2.2.1

ESPECIFICACIONES DE NOMBRES DEARCHIVOS.

- Máximo de 255 caracteres si se soportan nombres de archivos largos.
- Normalmente contiene caracteres:
  - Alfa-numéricos.
  - Caracteres especiales.

Figura 2.2.1 Especificación de nombre de archivos

La mayoría de los caracter ASCII puede ser utilizado en el sistema UNIX, excepto la barra inclinada (/) que tiene un significado especial, ya que actúa como un separador entre directorios y archivos dentro del Sistema UNIX. Los siguientes caracteres pueden ser utiliazados en nombres de archivos, pero es mejor evitarlos:

Especificaciones de nombres de archivos.			
¡!	Signos de admiración	**@**	Arrob
#	Signo de número	**$**	Signo de dólar

&	Ampersan	**^**	Signo de intercalación
( )	Paréntesis	**{ }**	Llaves
“ ”	Comillas simples o dobles	*	Asterisco
;	Punto y coma	**¿?**	Signos de interrogación
|	Cauce	****	Barra invertida
<>	Signos mayor o menor que		SPACEBAR
	TAB		BACKSPACE

Tabla 2.2.1 Caracteres evitados en nombres de archivos

2.2.2 Tipos de archivos

En el Sistema UNIX el punto (.) puede aparecer donde sea y en múltiples ocasiones en un nombre de archivo ejemplo: a.estu1.c.s..4... El punto es especial cuando aparece como el *primer carácter* de un nombre de archivo, a estos archivos se les llama *archivo oculto*. A continuación se definen algunos tipos de archivos en el Sistema UNIX y un ejemplo de cada uno de ellos, como se muestran al se desplegados con el comando *ls —l* como se muestra en la tabla 2.2.2:

Tipos de Archivos.
- Un archivo regular - r - - r- - r - - root /etc/passwd
d Un directorio drwxrwxr-x bin /usr/ucb
l Un archivo ligado simbólicamente lrwxrwxrwx root /dev/diskeette ⟶ ../devices/fd@1,f7200000:c
c Un archivo de dispositivo de carácter (terminales, impresoras)

crw-rw-rw- root /devices/zs@1,f1000000:a	
b	Un archivo de dispositivo de bloque (discos, CD, etc)
brw-rw-rw- root /devices/fd@1,f7200000:a	

Tabla 2.2.2 Tipos de archivos

2.3 Estructura jerárquica de los archivos

Debido a que los directorios pueden contener otros directorios, que a su vez pueden contener otros directorios, el sistema de archivos de UNIX es denominado *sistema de archivos jerárquicos*. Al sistema de archivos se le conoce como *estructura en árbol*, por que cada directorio permite bifurcar hacia otros directorios y archivos.

En la figura 2.3 se muestra una parte la estructura jerárquica del Sistema UNIX, es la parte fundamental y es instalado y configurado cuando se inicie una sesión. El Sistema UNIX también provee comandos que le permiten crear fácilmente nuevos archivos y directorios, así como para mover o copiar archivos de un directorio a otro, esto se verá más adelante.

ESTRUCTURA DE ARCHIVOS SISTEMA UNIX.

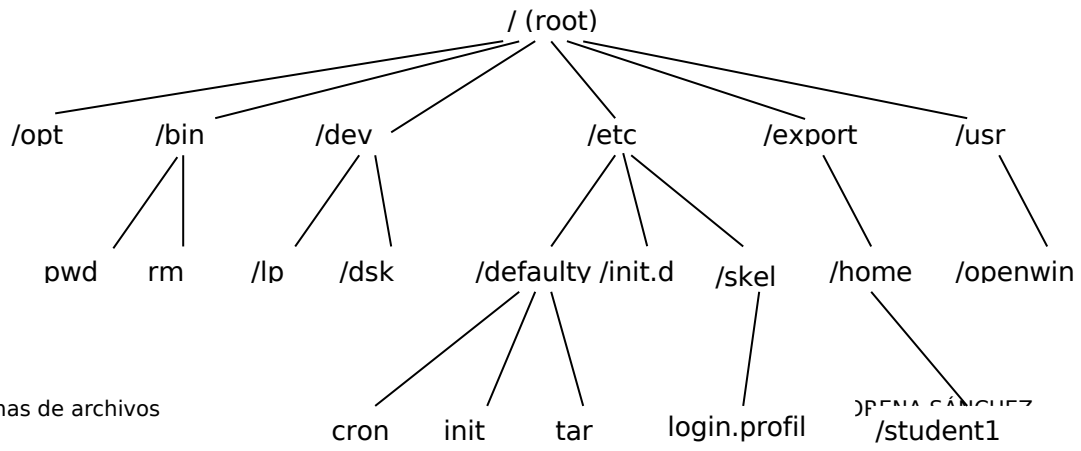


Figura 2.3 Estructura de archivos

2.3.1 Trayectorias (Path-names)

Una trayectoria, llamada también path-name, representa la ruta seguida en la jerarquía para llegar a un archivo o directorio. Cuando se indica la trayectoria de un archivo o directorio, se usa la diagonal o slash (/) para delimitar el directorio y/o nombres de archivos.

Ejemplo de trayectoria (/):

directorio/directorio/directorio
directorio/archivo

- En el siguiente ejemplo, el comando *ls -l* despliega la ruta del archivo passwd.

```
$ ls –l
- r-- r-- r--    root   /etc/passwd
```

Cuando un usuario entra al Sistema UNIX estará en algún directorio del árbol jerárquico. El usuario podrá cambiar su posición a algún otro directorio por medio de comandos, pero siempre estará en algún directorio.

La localización de archivos y directorios se puede indicar con una trayectoria absoluta o relativa.

Internamente, el Sistema UNIX localiza todos los archivos o directorios usando un nombre de trayectoria absoluta. Esto tiene sentido ya que la trayectoria absoluta identifica únicamente y de manera absoluta un archivo o directorio (ya que sólo existe una raíz "/root"). El sistema permite el uso de trayectorias relativas sólo como una conveniencia para el usuario.

2.3.1.1 Trayectoria absoluta

Una trayectoria absoluta especifica un archivo o directorio empezando desde el directorio raíz (/root). La siguiente figura 2.3.1.1. define una trayectoria absoluta.

TRAYECTORIA ABSOLUTA.

- Muestra la localización completa de un archivo o directorio.
- Siempre inicia en la parte superior de la jerarquía raíz (/root).
- Siempre inicia con la diagonal o slash (/).
- No depende de su localización actual en la jerarquía
- Siempre es única en la jerarquía entera.

Figura 2.3.1.1 Trayectoria absoluta

Ejemplo, Trayectoria absoluta:

- Las siguientes trayectorias indican la localización de todos los archivos llamados **archivo1** en la jerarquía. Se puede observar que hay muchos archivos con este nombre, pero cada uno de ellos tiene una trayectoria absoluta única. Usaremos el comando *find* para realizar esta tarea.

```
$ find / –name archivo1 -print
/tmp/ archivo1
/users/sem/ archivo1
/users/josel/ archivo1
/users/josel/memo/ archivo1
```

2.3.1.2 Trayectoria relativa

Una trayectoria relativa especifica un archivo en relación al directorio actual (donde se encuentra el usuario posicionado). En la figura 2.3.1.2 se define una trayectoria relativa:

TRAYECTORIA RELATIVA.

- Siempre empieza en su localización.
- Nunca inicia con una /
- Es única, relativa sólo a su localización actual.
- A menudo es más corta que la trayectoria absoluta.

Figura 2.3.1.2 Trayectoria relativa

Ejemplo, Trayectoria relativa:

- Las siguientes trayectorias indican la trayectoria relativa del archivo1.

  Supóngase que la posición actual es /users utilizamos el comando *find* para buscar los archivo1 en esta posición:

```
$ find  . –name archivo1 -print
sem/archivo1
josel/archivo1
josel/memo/archivo1
```

2.4 Navegando en el sistema de archivos

Este capítulo ayudará a comprender y manejar algunos comandos más usados para navegar en el sistema de archivos, como son: listar un archivo, crear y borrar un archivo y/o directorio, moverse entre directorios, etc.

2.4.1 ¿Qué podemos hacer con archivos y directorios?

Dado que muchas actividades del Sistema UNIX se enfocan a archivos y directorios, hay muchos comandos disponibles para manipularlos, en este tutorial veremos algunos de ellos. También se necesitará crear y manipular el contenido de un archivo, esto se hace comúnmente por medio de un editor tal como ***vi***, (ver Capítulo 3). Las siguientes tablas muestran algunos comandos para realizar estas funciones:

¿QUÉ PODEMOS HACER CON ARCHIVOS?

ls	Ver y listar las características de un archivo.
cat	Concatena y despliega el contenido de un archivo.
more	Ver el contenido de un archivo, una pantalla completa a la vez.
lp	Imprimir un archivo.
cp	Hacer una copia de un archivo o directorio.
mv	Cambiar el nombre de un archivo o directorio
mv	Mover un archivo a otro directorio.
ln	Crear otro nombre para un archivo (liga).
rm	Remover un archivo.

Tabla 2.4.1 Algunos comandos para el manejo de archivos

COMANDOS BÁSICOS DEL SISTEMA DE ARCHIVOS

pwd	Despliega el directorio actual del usuario.
ls	Despliega toda la lista de archivos y directorios.
cd	Cambia a un nuevo directorio, este comando acepta rutas absolutas o relativas.
find	Busca un o más archivo por su nombre en toda la jerarquía del sistema.
mkdir	Crea un directorio.
rmdir	Borra un directorio.

2

La principal función de la seguridad del sistema es mantener a los usuarios fuera de un acceso no autorizado, es decir, un usuario que no tenga ciertos

privilegios (permisos para leer, escribir y ejecutar) sobre un archivo o directorio no podrá trabajar con estos.

Conservar la información segura es lo más importante para el usuario así como para el administrador del sistema. Cada archivo es propiedad de un usuario del sistema y este tiene el control máximo sobre ellos.

El Sistema UNIX proporciona una estructura de acceso:

- Acceso de usuario
- Acceso de grupo
- Acceso de otros

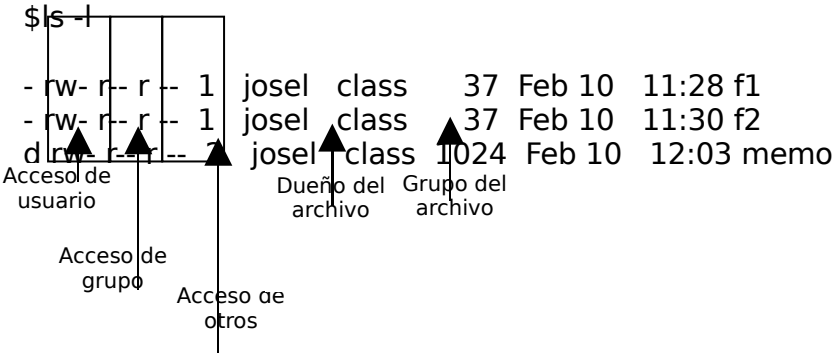
y cada acceso cuenta con tres clases de permisos:

- Permiso de lectura
- Permiso de escritura
- Permiso de ejecusión

Cuando se crea un archivo o directorio estos pertenecen automáticamente al usuario quien los creo. La figura 2.5 mustra estos tres accesos en recuadro y el dueño y grupo de los archivos:

PERMISOS Y ACCESOS

Los permisos se despliegan con el comando *ls –l*:



pueden ser fá

2.5.1 Tipos de accesos

Para manipular un programa o archivo se requiere que los archivos cuenten con ciertos permisos. Por ejemplo, para desplegar (*cat*) un archivo se requieren permisos de lectura en el archivo. De igual forma un directorio requiere permisos de lectura para listar su contenido con el comando (*ls*).

Los accesos depenen si es a un archivo o a un directorio y estos pueden proteger al archivo de ser o no borrado. Los permisos sólo pueden ser modificados por el propio dueño o por el superusuario.

La siguiente tabla muestra los tres accesos para un archivo o directorio:

Tipos de permisos:

Existen tres tipos de permisos para cada archivo y directorio

	Permisos	Archivo	Directorio	Algunos comandos permitidos
r	Lectura (Read)	El contenido puede se examinado.	El contenido puede se examinado.	more, cat, vi, cp, ln, ls, etc.
w	Escritura (Write)	El contenido puede se cambiado.	El contenido puede se cambiado.	vi, crear y borrar archivos.
x	Ejecución (Execute)	El archivo puede ser usado como un comando (programa).	Puede convertirse en el directorio de trabajo actual.	cd, si no tiene este permiso no se puede acceder a ese subdirectorio de trabajo.

El 2.2.2 comando chmod

El comando chmod se utiliza para cambiar los permisos de un archivo o un directorio, este soporta un método alfabético para definir los permisos de un archivo. En la siguiente figura se muestra el comando, la definición y manejo de estos permisos

A quien	
U	usuario
G	grupo
O	otros

Operador	
+	Adiciona
-	Sustrae
=	igual a

Octal	Permiso	
4	**r**	lectura
2	**w**	escritura
1	**x**	ejecución

COMANDO chmod

Sintaxis:

chmod [*lista_modos*] [*archivo*]

Lista de modos:

Figura 2.5.1.1 . Comando chmod y definición de permisos.

Hay dos formas de cambiar los permisos: una llamada forma simbólica y otra de opción octa. Para ello se utiliza el comando chmod que acontinuación se detalla.

- **chmod modo simbólico**

El modo simbólico usa combinaciones de letras y símbolos para sumar o quitar permisos al archivo según el acceso (ver *Figura 2.5.1.1* Comando chmod y definición de permisos).

Ejemplo chmod modo simbólico: El archivo f1

- Permisos Originales f1

```
$ ls –l f1
-rw- rw- r--  1  estu1 admon   Jul  16 11:50   f1
```

- Quitar permisos de escritura al grupo

```
$ chmod g - w  f1
-rw- r-- r--  1   estu1   admon     Jul  16 11:52 f1
```

- Adicionar permiso de ejecución al usuario, al grupo de lectura y escritura y ningún permiso a otros.

```
$ chmod u+x, g=rw, o-r    f1
-rwx rw ---  1 estu1      admon      Jul    16
11:56f1
```

- Para deshabilitar todos los permisos de un archivo use el comando:

```
$ chmod =      f1
- --- --- ---    1 estu1  admon      Jul        16
11:58  f1
```

Octal	Permiso
7	**r w x**
6	**r w -**
5	**r – x**
4	**r - -**
3	**- w x**
2	**- w -**
1	**- - x**
o	**- - -**

▪ chmod modo octal

El modo octal se refiere al uso absoluto o modo numérico para representar los permisos de un archivo. En la siguiente figura se muestran las ocho combinaciones posibles para colocar los permisos a un archivo. Esto es una combinación de tres permisos para el usuario, otros tres para el grupo y finalmente tres para otros:

PERMISOS MODO OCTAL

Octal	Permiso	
4	**r**	lectura
2	**w**	escritura
1	**x**	ejecución

Figura.2.5.1.1.1 Permisos modo octal

Ejemplo chmod modo octal:

- Permisos Originales de f1:

```
$ ls –l f1
-rw- rw- r--      1      estu1 admon      Jul  16 11:50   f1
```

- Quitar permisos de escritura al grupo:

```
$ chmod 644 f1
-rw- r-- r--   1   estu1 admon      Jul  16 11:52   f1
```

- Adicionar permiso de ejecución al usuario, al grupo de lectura y escritura y ningun permiso a otros:

```
$ chmod 760 f1
-rwx rw ---      estu1 admon      Jul  16 11:56   f1
```

- Para deshabilitar todos los permisos de un archivo use el comando:

```
$ chmod 000  f1
- --- --- --- 1   estu1 admon      Jul  16 11:58   f1
```

2.5.1.2 Comando chown

El dueño de un archivo se identifica con el usuario al que pertenece este archivo. Cuando se crea un archivo o directorio este o estos pertenecen al usuario que los creo.

Sólo el dueño del archivo o el superusuario pueden cambiar el propietario de un archivo. El comando chown se utiliza para realizar esta actividad como lo muestra la siguiente figura:

COMANDO chown - **Cambia el dueño de un archivo**.

Sintaxis:

chown [*usuario*] [*archivo*]

Figura 2.5.1.2 Sintaxis comando chown

Ejemplo chown:

```
$ id
```

uid = 101 (admon1), gid = 300 (clase1)
$ ls –l archivo*1*
-rwx rw- r-- 1 staff case1 25 Jul 6 18:38 archivo1

- Cabiamos a usuario staff o root

$ su – root
passwd:
chown admon1 archivo*1*
ls –l archivo1
-rwx rw- r-- 1 admon1 clase1 25 Jul 6 18:38 archivo1

2.5.1.3 Comando chgrp

El comando chgrp cambia el grupo del archivo o directorio al que pertenece. Este sólo puede ser cambiado por el dueño del archivo o el superusuario. El comando *chgrp* no funcionará si el nuevo grupo especificado no existe. La siguiente tabla muestra la sintaxis del comando chgrp:

COMANDO chgrp - **Cambia el grupo de un archivo**.

Sintaxis:

chgrp [grupo] [*archivo*]

Figura 2.5.1.3 Sintaxis comando chgrp

Ejemplo chgrp:

```
$id
uid = 101 (admon1),  gid = 300  (clase2)
$ ls –l  archivo1
-rwx rw- r--  1  admon1    clase1   25  Jul 6 18:38   archivo1
$chgrp clase2 archivo1
$ls –l archivo1
 -rwx rw- r--  1  admon1   clase2   25  Jul 6 18:38   archivo1
```

2.5.2. Comando umask

Cuando se crea un archivo o directorio los permisos son asignados automáticamente por el sistema: para un archivo se utilizan la mascara 666 y para un directorio 777. Para proteger los archivos que se crean se puede usar el comando *umask*, este comando permite especificar los permisos de todos los archivos que se crean después de emitir la orden umask. Este comando puede ser utilizado en línea o puede ser incluido dentro del profile del usuario para dejarlo fijo, así que, cada que se inicie una sesión con este usuario y genere un archivo o directorio estos tendrán los permisos especificados en el umask.

Ejemplo parte del contenido del archivo .profile:

```
TERM= vt100        Trabajar con la terminal vt100.
export term
calendar        Mostrar el calendario al iniciar la sesión.
umask= 022
```

La especificación de permisos con *umask* no es complicada, se simplifica si se recuerdan estos puntos:

- *umask* utiliza un código numérico para representar permisos de la misma forma que *chmod octal.*
- Se especifican los permisos diciéndole a *umask* que *reste* de los permisos el valor que no se requiere por acceso.

La siguiente tabla recuerda la forma octal del chmod para que sea utilizada en el comando umask:

Octal	**Permiso**	
4	**r**	lectura
2	**w**	escritura
1	**x**	ejecución

Tabla 2.7 Comando chmod forma octal.

La siguiente tabla muestra la función y ejemplos del comando *unmask*:

Valores de umask en an archivo			
Permisos	umask	Resultado de permisos en el archivo	Descripción
7 7 7	umask = 0 0 0	- rwx rwx rwx	Todos los permisos
6 6 6	umask = 1 1 1	- rw- rw- rw-	Se quitan permisos de ejecución para los tres accesos
7 5 5	umask = 0 2 2	- rwx rw- rw-	Se quitan permisos de escritura para los dos accesos: grupo y otros

Tabla 2.5.2 Comando umask función y ejemplos

Ejemplo umask:

- Permisos automáticos de un archivo = 6 6 6

```
$ ls –l archivo1
- rw- rw- rw-   12 estu1 admon3     Jul  16 11:56   archivo1
```

- Se requieren permisos = 6 4 4

A la máscara se le restan los permisos de escritura como lo muestra la *Tabla 2.5.2* Comando *umask* función y ejemplos:

```
$ umask = 133
```

Al crear el archivo1 se verifican los permisos:

```
$ ls –l archivo1
- rw- r- - r- -   12    estu1    admon3     Jul  16 11:56   archivo1
```

Ejemplo umask:

- Verificar la mascara actual:

```
$ umask
0022
```

- Cambiar la mascara a 027

```
$ umask 027
$ umask
0027
```

2.6. Impresión de un archivo

El Sistema UNIX incluye la funciónalidad de imprimir archivos y documentos. Se puede utilizar para imprimir desde archivos de texto simples hasta grandes documentos con formatos complejos. Proporciona una interfaz simple y uniforme a una amplia variedad de impresoras. En éste tutorial no se verá la configuración y mantenimiento de las impresoras, sólo se analizarán los camandos básicos para imprimir.

El sistema lp es en sí mismo grande y complejo, pero afortunadamente su complejidad está oculta a los usuarios. De echo, las tres órdenes básicas son: *lp*, *lpstat* y *cancel*.

2.6.1 Comando lp

El comando *lp* permite al usuario enviar archivos a imprimir. Un número único de identificación de trabajo (llamado un request ID) es dado a cada archivo envíado. El *request ID* se puede utilizar para comprobar el estado del trabajo o para cancelarlo si se desea. En la siguiente figura se muestra la sintaxis del comando lp:

COMANDO lp

- Envía un archivo a la cola de impresión.
- Asigna un número ID único.

Sintaxis.

lp [-argumento] [*nombre-archivo*]

Figura 2.6.1 Sintaxis comando lp

La siguiente tabla muestra los argumentos del comando lp:

lp		
ARGUMENTO	**SINTAXIS**	**COMENTARIO**
-d	# lp –d printername	(printername) Nombre de la impresora en la cual será impresa la solicitud.
-n	# lp –n num	Imprime un número de copias de la solicitud. Por default es uno.
-t	# lp –t título	Imprime el título en la página banner de la impresora. La página banner es la página del encabezado que identifica el dueño de la impresión.

Tabla 2.6.1 Argumentos comando lp

Ejemplo lp:

- Imprimir el archivo reporte1 :

```
$lp reporte1
request id is print1-112 (1 file)
```

- Se pueden imprimir varios archivos de una vez incluyendo todos en el argumento:

```
$lp repor*
request   id   is    print1-154    (4 file)
```

- Se utiliza *lp* para imprimir directamente la salida de una orden (salida estándar)

```
$ls  -l | lp
```

2.6.2 Comando lpstat

El comando *lpstat* reporta el estado de los trabajos de impresión, las impresoras disponibles en el sistema y el número de trabajos enviados. Uno de los usos más imporantes del comando lpstat consiste en ver si un trabajo de impresión esta siendo atendido o si existe algun problema. La figura 2.6.2 muestra la sintaxis del comando lpstat:

COMANDO lpstat.

- Estado del trabajo de impresión.
- Impresoras disponibles y su estado.

Sintaxis:

lpstat [- argumento]

Figura 2.6.2 Argumentos comando lpstat

Ejemplo lpstat:

```
$ lpstat
print1-142   reporte1  1730   Aug  20  12:09    on print1
laser2-136   reporte2     56   Aug  20   12:17
```

El primer trabajo se esta imprimiendo en la impresora print1 y el segundo se encuentra en espera para ser enviado a la impresora laser.

2.6.3 Comando cancel

A veces se necesita cancelar un trabajo de impresión, la orden *cancel* permite detener cualquiera de los trabajos de impresión, incluso los que actualmente se están imprimiendo. La siguiente figura muestra la sintaxis del comando cancel.

COMANDO cancel.

Los argumentos para el comando *cancel* pueden ser de dos tipos:

- Un request ID (Se obtiene con el comando lp).
- Un nombre de impresora.

Sintaxis:

cancel [request-ID] [printer]

Figura 2.6.3 Argumentos comando cancel

Ejemplo cancel:

```
$ lpstat
print1-142   reporte1  1730   Aug  20  12:09    on print1
laser2-136   reporte2    56   Aug  20   12:17

$ cancel  print1-142
request “print1-142” cancelled
```

CAPÍTULO 3. Introducción al Editor vi y Fundamentos de shell

3.1 ¿Qué es el vi?

Vi es un editor de texto estándar que es proporcionado con la mayoría de los Sistemas UNIX. Un editor de texto es un programa de computadora interactivo que le permite escribir o modificar texto de un archivo. Se puede utilizar el *vi* para crear nuevos archivos o alterar los existentes. Algunas características del editor *vi* se mencionan en la siguiente figura:

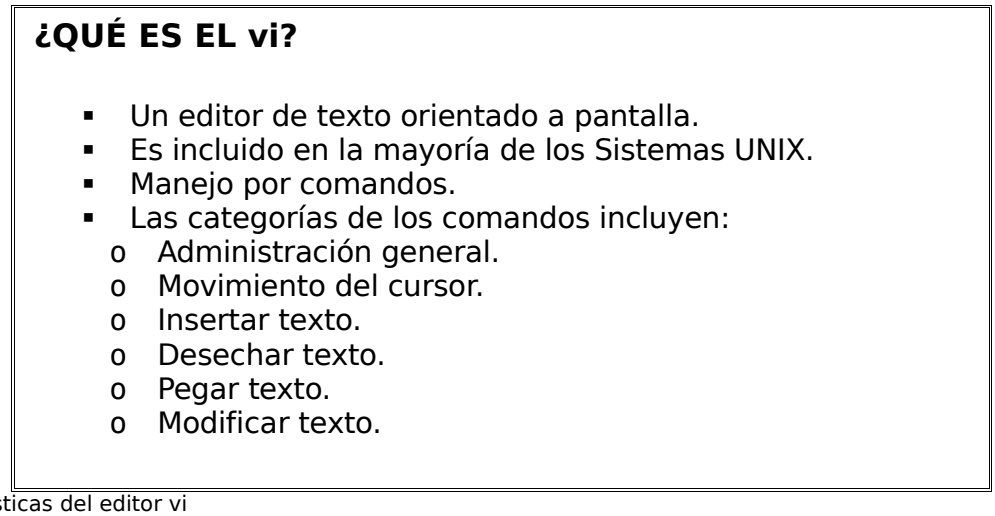


Figura 3.1

Características del editor vi

3.1.1 Iniciando una sesión de vi

La siguiente figura muestra la sintaxis para iniciar el editor *vi*.

COMANDO vi
Sintaxis:

vi [*nombrearchivo*]	Abre o crea un archivo.
vi	Abre el editor de texto.
vi – r	Recobra un archivo dañado.
view [*nombrearchivo*]	Abre un archivo de sólo lectura.

- Si el archivo ya existe, se desplegará el contenido en la pantalla.
- Si el archivo es nuevo se verá un pantalla con una columna de tildes (~) en la parte izquierda.

Figura 3.1.1 Sintaxis para inicial el editor vi

El ejemplo siguiente muestra la edición de un archivo nuevo:

Ejemplo del editor *vi*:

```
$ vi archivo1
~
~
~
~
~
~
~
~
```

3.1.1.1 Configuración de la terminal

Debido a las diferentes características de las terminales, lo primero que se debe hacer antes de utilizar el editor *vi,* es especificar el tipo de terminal. Si se encuentra una terminal mal configurada el editor *vi* se comportará de manera extraña (desplazamiento del texto, mala visualización del texto, etc.). Existe una gran variedad de terminales, por tal motivo, sólo se vera el tipo de terminal más usada.

El comando *env,* que veremos más adelante, despliega todas las variables que se encuentren configuradas en el momento que un usuario inicie su sesión.

Ejemplo Configuración de TERM:

```
$ env
EDITOR=/usr/bin/vi
HOME=/export/home/estu1
HZ=100
LOGNAME= estu1
MAIL=/var/mail/estu1
OPENWINHOME=/usr/openwin
PATH=:/bin:/usr/bin:/usr/sbin:/usr/openwin/bin:/export/home/estu1:.
PS1=merton1
SHELL=/bin/sh
TERM=sun-cmd
TZ=MEX/Central
$
```

Para cambiar la terminal del ejemplo anterior se cambia la variable ‘TERM’ en el archivo .profile del usuario o se puede hacer en línea de comando como a continuación se muestra:

```
$TERM=vt100
$export TERM
```

3.1.1.2 Modos del editor vi

El editor *vi* contiene dos modos de comandos:

Modo entrada: Cuando el editor está en modo de entrada, los caracteres que se tecleen, se introducen simplemente como texto.

Modo orden: En el modo de orden (comandos del editor *vi*), los caracteres que se tecleen, son órdenes que controlan el texto o ejecuciones de algún comando para manipular el texto.

Cuando se encuentra en modo de entrada de texto, la tecla ESC (escape) lo cambia al modo de orden.

En el modo orden, se puede ejecutar algún comando propio del editor *vi*, para ello es necesario después de oprimir la tecla ESC teclear dos puntos ":" y después introducir el comando que será ejecutado:

Ejemplo ejecución de comandos del editor vi:

```
$ vi archivo1
Hola mundo
Hola mundo
Hola mundo
Hola mundo
~
~
~
~
:2d
```

Al teclear los dos puntos ":", el cursor se colocará en la parte inferior de la pantalla, enseguida se introducen el o los comandos (más adelante se describirán algunos de ellos) que serán diferenciados en este tutorial por los dos puntos al principio del comando.

3.1.2 Introducción del modo de entrada

El editor *vi* es manejado por ordenes (comandos). Estas ordenes se usan comúnmente para modificar y manipular el texto de una o más líneas a la vez (multilínea). Para poder introducir texto en su archivo, debe estar en modo de entrada. Muchas ordenes del editor *vi* son una abreviación de algún significado asociado, esto facilita el manejo de las órdenes. A continuación se muestran las órdenes para manipular texto dentro del editor *vi*:

Comando	Significado
a	Adiciona texto después del cursor.
A	Adiciona texto nuevo al final de la línea.
o	Abre una línea para texto abajo de la línea actual.
O	Abre una línea para texto arriba de la línea actual.
i	Inserta texto nuevo antes del cursor.
I	Inserta texto nuevo al inicio de la línea.

Tabla 3.1.2 Ordenes para manipular texto

3.1.3 Comandos de control del cursor

Una de las ventajas principales del editor *vi* es el manejo de pantallas, esto es, que se pueden ver partes del texto (como si se manejarán páginas), además es posible moverse a través del texto: con ordenes del editor vi hacia arriba, hacia abajo y a los lados como se muestra en la tabla 3.1.3 o usar el teclado para desplazarse dentro del texto como se muestra en la tabla 3.1.3.1.

Comando	Significado
W	Moverse a la siguiente palabra o signo de puntuación.
w	Moverse a la siguiente palabra.
B	Moverse hacia atrás palabra por palabra o signo de puntuación.
b	Moverse hacia atrás palabra por palabra.
E	Mueve el cursor al final de la siguiente palabra.
G	Ir al final del archivo
#G	Ir a la línea número #.
: #	Ir a la línea número #.
Ctrl + g	Reporta la línea donde esta colocado.
Ctrl + b	Corre hacia atrás a la ventana anterior del texto.
Ctrl + f	Corre hacia adelante a la ventana siguiente del texto.
Ctrl + u	Corre hacia arriba a la mitad de la ventana de texto.
Ctrl + d	Corre hacia abajo a la mitad de la ventana de texto.
H	Mueve el cursor al principio del texto.
M	Mueve el cursor a la mitad de todo el texto.
L	Mueve el cursor al final de todo el texto.

Tabla 3.1.3 Ordenes de desplazamiento en editor vi

En la siguiente figura se muestran las teclas utilizadas para desplazarse dentro del texto:

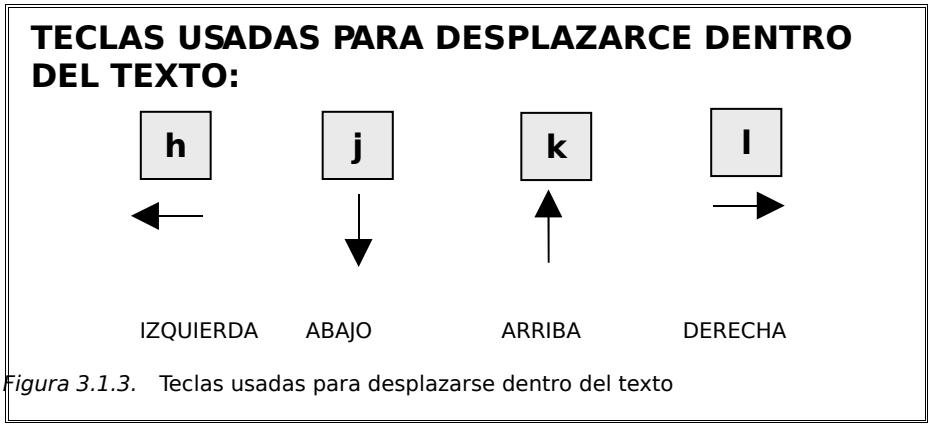


Figura 3.1.3. Teclas usadas para desplazarse dentro del texto

3.1.4 Terminando una sesión de vi

Al terminar de trabajar con el texto, será necesario salvar el contenido, en la siguiente tabla se muestran algunos comandos para realizar esta tarea:

Comando	Significado
zz	Salva el archivo y sale del editor *vi*. (No se teclean dos puntos ":").
:w	Salva los cambios.
:wq	Salva los cambios y sale del editor *vi*.
:wq!	Salva los cambios incondicionalmente (aunque el archivo sea de sólo lectura) y sale del editor *vi* .
:w nombredearchivo	Salva los cambios realizados con el nuevo nombre asignado.
:q!	Sale del editor *vi* sin salvar los cambios.

Tabla 3.1.4 Ordenes del editor vi para guardar archivos

El comando seguido del signo '**!**' es incondicional, o sea, que realizará el comando sin importar los permisos que tenga el archivo.

3.1.5 Manejo del texto

El editor *vi* ofrece un gran gama de opciones para su uso y con ello, facilita el manejo del texto como es: copiar, pegar, sustituir, etc. A continuación se verán algunas de estas tareas.

3.1.5.1 Copiar y pegar texto

Para copiar y pegar texto dentro del editor *vi*, se pueden utilizar las siguientes opciones:

Comando	Significado
yy	Copia la línea donde se encuentra posicionado el cursor.
yG	Copia las líneas desde el cursor hasta el final del archivo.
y$	Copia desde el cursor hasta el final de la línea.
:1,3 co 5	Copia de la línea 1 a la 3 y la pega después de la línea 5.
p	Pega la líneas copiadas por cualquier comando de copiado, arriba de la línea en la que se encuentra el cursor.
P	Pega la líneas copiadas por cualquier comando de copiado, de bajo de la línea en la que se encuentra el cursor.

Tabla 3.1.5.1 Ordenes del editor vi para copiar y pegar texto.

3.1.5.2 Borrar texto

Para borrar texto dentro del editor *vi*, se utilizan las siguientes opciones:

Comando	Significado
x	Borra el carácter donde se encuentra el cursor.
X	Borra el carácter a la izquierda del cursor.
dw	Borra la palabra o parte de la palabra después del cursor.
#dw	Borra el numero de palabras indicadas por el #.
dd	Borra la línea donde se encuentra el cursor.
#dd	Borra el número de líneas indicadas por el #, donde se encuentra el cursor hacia abajo.
D	Borra el contenido de la línea donde se encuentra el cursor dejando la línea en blanco.
dG	Borra las líneas desde el cursor hasta el fin del archivo.
D1G	Borra las líneas desde el cursor hasta el principio del archivo.
:5,10d	Borra de la línea 5 a la 10.

Tabla 3.1.5.2 Ordenes del editor vi para borrar texto.

3.1.5.3 Cambiar texto

Para cambiar el texto dentro del editor *vi*, se puede utilizar las siguientes órdenes o comandos:

Comando	Significado
cw	Cambia la palabra actual.
#cw	Cambia el número de palabras indicadas por el #.
r	Remplaza el carácter donde se encuentra el cursor.
R o C	Remplaza toda la línea (pasa a modo de sobre escritura)
S	Remplaza el carácter donde se encuentra el cursor por una cadena.
r ENTER **i ENTER**	Rompe la línea y la pasa abajo.
J	Une la línea de abajo del cursor, subiéndola donde se encuentra el cursor.
Xp	Traslada el carácter donde se encuentra el cursor a la derecha del siguiente caracter.
~	Cambia un carácter por mayúscula o minúscula.

Tabla 3.6 Ordenes del editor vi para cambiar texto

3.1.5.4 Deshacer o repetir una orden o comando

Para deshacer un comando (undo) dentro del editor *vi*, se puede utilizar la información de la siguiente tabla:

Comando	Significado
u	Deshace el comando anterior.
U	Deshace todos los comandos realizados en la línea.
:u	Deshace el comando anterior realizado en la última línea.
.	Repite el comando anterior.

Tabla 31.5.4 Ordenes del editor vi para deshacer una orden o comando

3.1.6 Comandos avanzados de vi

El editor *vi* ofrece la utilización de patrones, los de búsqueda y reemplazo, así como concatenar varios archivos en uno o introducir al editor *vi* comandos shell (comandos del Sistema UNIX: date, ls, etc), esto facilita la utilización del editor *vi*.

3.1.6.1 Insertar texto o un comando shell

Para insertar comandos shell o texto que se encuentran fuera de un archivo abierto con el editor *vi*, se puede manejar con los siguientes comandos:

Comando	Significado
:r archivo	Inserta un archivo externo al archivo abierto con el editor *vi*, debajo del cursor.
:# r archivo	Inserta un archivo externo al archivo con el que se esta trabajando en la línea indicada por #.
:r !cmd	Inserta una orden de shell en el archivo por ejemplo: :r !date :# r !cal

Tabla 3.1.6.1 Ordenes del editor vi para insertar texto o un comando shell

3.1.6.2 Busca y remplaza texto

Se puede buscar y/o remplazar texto dentro de un archivo abierto con el editor *vi*. Cuando se busca el texto o palabra (cadena) al ser encontrada por la orden, el cursor se coloca en la posición donde se encontró dicha cadena. Se tienen las opciones que a continuación se muestran para realizar la búsqueda o reemplazo de texto:

Comando	Significado
/cadena	Busca el texto de la cadena en el archivo.
?cadena	Busca el texto de la cadena en el archivo desde la parte inferior.
n	Busca la siguiente cadena en el texto y lo realiza hacia abajo del texto.
N	Busca la siguiente cadena en el texto y lo realiza hacia arriba del texto.
M y n	Define las líneas sobre las cuales debe ser ejecutado el comando: :5,8 s/uno/dos
:%s/texto_viejo/texto_nuevo/g	Remplaza la palabra o palabras vieja por la nueva en todo el texto (globalmente).

Tabla 3.1.6.2 Ordenes del editor vi para buscar y remplazar texto dentro de un archivo

3.1.6.3 Comando set del editor vi

Existen muchas comandos disponibles dentro del editor *vi*, las cuales pueden hacer más facil el manejo del texto dentro, una de ellas es el comando *set* que contiene varias opciones para el manejo global del texto, es decir, aplicará la instrucción señalada (opciones) en todo el texto.

El comando set se utiliza dentro del editor *vi*. El comando set puede ser activado o desactivado fácilmente como se muestra enseguida:

La sintaxis para activar (**on**) una opción es:

:set *opción*

La sintaxis para desactivar (**off)** una opción es:

:set no*opción*

La lista de todas las opciones de set y su valor se puede desplegar con:

:set all

Algunas opciones del comando set se muestran en la siguiente tabla:

Comando	Significado
:set number	Enumera todas las líneas.

:set wm=15	Las líneas se dividirán automáticamente 15 espacios antes del borde de la pantalla.
:set list	Muestra caracteres invisibles encontrados con tab y fin de línea.

Tabla 3.1.6.3 Opciones del comando set

Ejemplo comando set del editor vi:

- En este ejemplo se utiliza la opción “number” que enumera toda la líneas, las peticiones con el comando set se realizan inmediantamente en el texto

```
$ vi archivo1
Hola mundo
Hola mundo
Hola mundo
Hola mundo
~
~
~
~
:set number
```

Resultado:

```
$vi archivo1
1    Hola mundo
2    Hola mundo
3    Hola mundo
4    Hola mundo
~
~
~
~
```

El comando set se puede utilizar con cualquiera de sus opciones, se pueden establecer automáticamente cada vez que el usuario ingrese al editor *vi*, ya que al salir del texto, aunque se alla salvado el archivo se pierde lo que se trabajo con el comando set, esto es posible colocando una línea en un archivo creado por el usuario llamado *.exrc* en el directorio HOME (directorio del usuario):

Ejemplo del contenido del archivo .exrc:

set *number* (Sin puntos al principio del comando)

3.2 Fundamentos de shell

Este apartado abarcará algunos puntos importantes para aprender el manejo básico del interprete shell, como son: analizar que ocurre al entrar al sistema, describir las tareas del shell, entender y manipular variables de ambiente, personalizar y analizar las funciones de las variables de ambiente de un usuario o de una aplicación.

3.2.1 ¿Qué es un shell?

Un shell es un programa interactivo que sirve como un intérprete de líneas de comandos (El shell es el que comunica al usuario con el Sistema UNIX). Una tarea del shell es permitir que se introduzcan comandos y pasarlos al sistema operativo para su ejecución. Los siguientes puntos resumen la funcionalidad del shell:

- Busca un comando y ejecuta el programa asociado.
- Sustituye valores de variables.
- Realiza sustitución de comandos.
- Completar nombres de archivos a partir de caracteres, para generar nombres de archivos completos.
- Proporciona una interfaz de programación.

3.2.2 Shell comúnmente usados

El shell nos ayuda a personalizar nuestro ambiente de trabajo. En este apartado se analizarán los tres tipos de shell más comúnes: sh, ksh y csh. La siguiente tabla identifica las características y alcances de trabajo, de cada shell en el Sistema Operativo UNIX Solaris:

CARACTERISITCAS	**BOURNE** Sh	**C** csh	**KORN** ksh
Alias	**No**	**Si**	**Si**
Edición de línea	**No**	**No**	**Si**
Cambiar de directorio	**Si**	**Si**	**Si**
Historial de comandos	**No**	**Si**	**Si**
Control de tareas	**No**	**Si**	**Si**
Shell restringidos	**Si**	**No**	**Si**
Proteger archivos de sobre-escritura	**No**	**Si**	**Si**

Tabla 3.2.2 Características de shell

3.2.3 El ambiente de usuario

El ambiente de usuario es el que limita el entorno de trabajo del usuario, algunos ejemplos se muestran a continuación:

- El directorio propio del usuario.
- Definir el nombre del usuario.
- Define la impresora, si es que existe.
- Puede fija el historial de comandos.
- Avisa si tiene correo.
- Tipo de shell.
- Mensajes de bienvenida.

Esto sólo es una pequeña parte de las tareas que se pueden desarrollar dentro del ambiente de usuario, algunas de estas, se vieron en los capítulos anteriores: como el tipo de shell, configuración de variables, etc. Al avanzar en este capítulo, se facilitará el entendimiento y la manipulación de muchas más configuraciones.

3.2.4 Estableciendo variables de shell

Otro servicio del interprete shell, es el mantenimiento de variables, las cuales forman parte del ambiente del usuario. Estas variables ayudan a obtener los recursos necesarios para el trabajo del usuario, existen dos tipos de variables: variables de ambiente y variables locales, a continuación se describen estos dos tipos de variables y los recursos que aporta cada una de ellas al usuario.

3.2.4.1 Variables de ambiente (env)

Las variables de ambiente, algunas veces referenciadas como variables globales, son variables fijas en el archivo .profile o en el .login, estas son leídas una vez que se ingresa al sistema. Las variables de ambiente son variables predefinidas por el sistema, algunos ejemplo son: TZ, LOGNAME y PATH.

El comando *env* despliega las variables de ambiente y sus valores, del usuario con el que se esta trabajando:

Ejemplo comando env:

```
$ env
EDITOR=/usr/bin/vi
HOME=/export/home/estu1
HZ=100
LOGNAME= estu1
MAIL=/var/mail/estu1
OPENWINHOME=/usr/openwin
PATH=:/bin:/usr/bin:/usr/sbin:/usr/openwin/bin:/export/home/estu1:.
PS1=merton1
SHELL=/bin/sh
TZ=MEX/Central
$
```

3.2.4.2 Variables locales (set)

Las variables locales, son variables creadas en línea por el usuario y son interpretadas sólo en la sesión en la que se ingresaron. Estas variables se pierden cuando el usuario sale de la sesión.

Con el comando *set* se despliegan las variables locales y sus valores, del usuario con el que se esta trabajando:

Ejemplo comando set:

```
$ set
EDITOR=/usr/bin/vi
HOME=/export/home/estu1
HZ=100
LOGNAME= estu1
MAIL=/var/mail/estu1
OPENWINHOME=/usr/openwin
PATH=:/bin:/usr/bin:/usr/sbin:/usr/openwin/bin:/export/home/estu1:.
PS1=merton$
SHELL=/bin/sh
TERM=sun-cmd
DIR=pwd
PEPE=/user/pepe/pruebas
TZ=MEX/Central
$
```

La siguiente figura muestra la sintaxis para ingresar o cambiar una variable local:

Sintaxis para obtener una variable local

$ *variable*=valor

Quitar la variable

$ unset *variable*

Figura 3.2.4.2 . Sintaxis de una variable local

Para que el shell interprete la variable es necesario exportarla:

Exportar variables

Sintaxis

$ *variable*=valor
$ export *variable*

Figura 3.2.4.2.1 Exportar una variable

Ejemplo variable local:

- En el siguiente ejemplo se añadirá al ambiente la variable TERM

```
$TERM= sun-cmd
$export TERM
```

- Se verifica el ingreso de la variable TERM en el ambiente con el comando *set*.

```
$ set
EDITOR=/usr/bin/vi
HOME=/export/home/estu1
HZ=100
LOGNAME= estu1
MAIL=/var/mail/estu1
OPENWINHOME=/usr/openwin
PATH=:/bin:/usr/bin:/usr/sbin:/usr/openwin/bin:/export/home/estu1:.
PS1=merton$
SHELL=/bin/sh
DIR=pwd
PEPE=/user/pepe/pruebas
TZ=MEX/Central
TERM=sun-cmd
```

- Ahora se cambiará el valor de variable TERM en el ambiente

```
$TERM=vt100
$export TERM
```

- Se verifica el nuevo valor de la variable TERM en el ambiente nuevamente con el comando *set*.

```
$ set
EDITOR=/usr/bin/vi
HOME=/export/home/estu1
HZ=100
LOGNAME= estu1
MAIL=/var/mail/estu1
OPENWINHOME=/usr/openwin
PATH=:/bin:/usr/bin:/usr/sbin:/usr/openwin/bin:/export/home/estu1:.
PS1=merton$
SHELL=/bin/sh
DIR=pwd
PEPE=/user/pepe/pruebas
TZ=MEX/Central
 TERM=vt100
```

- La siguiente tarea será quitar la variable TERM del ambiente

```
$ unset TERM
```

- Se verifica la inexistencia de la variable TERM en el ambiente con el comando *set*.

```
$ set
EDITOR=/usr/bin/vi
HOME=/export/home/estu1
HZ=100
LOGNAME= estu1
MAIL=/var/mail/estu1
OPENWINHOME=/usr/openwin
PATH=:/bin:/usr/bin:/usr/sbin:/usr/openwin/bin:/export/home/estu1:.
PS1=merton$
SHELL=/bin/sh
DIR=pwd
PEPE=/user/pepe/pruebas
TZ=MEX/Central
```

3.2.5 Comando which

La función del comando *which* es buscar el archivo ejecutable (código fuente) de algún comando, pero sólo busca en los directorios del sistema operativo. El comando *which* despliega la trayectoria absoluta donde se encuentra el archivo ejecutable del comando.

En ocasiones ocurre que al querer utilizar un comando, el sistema nos envía que no lo encuentra, esto es por que en el ambiente del usuario no se cuenta con la variable PATH (variable de trayectorias). El comando *which* es de gran utilidad por que al mostrar la trayectoria del archivo ejecutable podemos ejecutar el comando y a su vez colocarla en la variable de ambiente.

Ejemplo comando which:

- Cuando requerimos utilizar el comando *more (*el comando *more:* despliega el contenido de un archivo y lo realiza por pantalla, lo que lo hace muy útil para desplegar archivos de gran tamaño) nos índica el sistema que no lo encuentra, como lo muestra el siguiente ejemplo.

```
$ more  archivo1
Command no found
```

- Ejecutamos el comando *which* para encontrar el comando *more*:

```
$ which more
/usr/bin/more
```

- Ahora podemos utilizar el comando *more*, y podemos ver el contenido del archivo1:

```
$ /usr/bin/more archivo1
Hola mundo
Hola mundo
$
```

- Añadimos la trayectoria encontrada en el archivo .profile del usuario con el que se esta trabajando, para ello utilizamos el editor *vi.* En este ejemplo se mostrará sólo como debe quedar la variable PATH:

PATH=:/bin:/usr/sbin:/usr/openwin/bin:/export/home/estu1**:/usr/bin**:.

La sintaxis de la variable PATH es: Los campos son divididos por dos puntos ":", y al final se colocan dos puntos seguidos de un punto ":."

3.2.6 Comando alias

Un *alias* es darle un nombre nuevo a un comando. *Alias* puede ser muy útil por que es un método por el cual se abrevian líneas de comando largas. Los *alias* son usados frecuentemente como una forma corta para trayectorias completas. En la siguiente figura se muestra la sintaxis del comando *alias*:

COMANDO alias

Sintaxis:

$ alias [*nombre = cadena]*

Figura 3.2.6 Sintaxis del comando alias

Un *alias* estará disponible hasta que salga de la sesión, a menos que se coloquen en el archivo .profile del usuario. Algunos usuarios encuentran las características del comando *alias* tan flexibles que hace que la interfaz del Sistema UNIX utilice el nombre de algunos comandos que usualmente se manejan en otros sistemas como *Msdos, etc*.

Ejemplo comando alias:

- En el siguiente ejemplo se realizan tres *alias*:

  El primero *alias* cambia el comando *cd* con el nombre *pasa*.
  El segundo *alias* cambia una trayectoria larga por un nombre corto.
  El tercer *alias* cambia el comando *pwd* por el nombre *dir*.

```
$ alias  pasa =´cd ´
$ alias  pepe=/users/pepe/pruebas/depto1/caso1
$ alias  dir= ´pwd´
```

- Ahora se utilizan los comandos con su nuevo nombre: Primero se verifica el directorio actual con *dir*, después se cambia con *pasa* al directorio *pepe* y se verifica el directorio con *dir*. Por último se compara el uso de los comandos originales con los comandos *alias*:

Comandos originales	Comandos *alias*
$ pwd	$ dir
/users/lalo	/users/lalo
$ cd /users/pepe/pruebas/depto1/caso1	$ pasa pepe
$ pwd	$ dir
/users/pepe/pruebas/depto1/caso1	
/users/pepe/pruebas/depto1/caso1	

- Para verificar el valor de un *alias* en particular, lo hacemos de la siguiente forma:

```
$ alias  pasa
pasa =´cd ´
```

- También podemos ver todos los *alias* existentes de la siguiente forma:

```
$ alias
  cambiar=´cd ´
pepe=/users/pepe/pruebas/depto1/caso1
dir=pwd
```

- Para borrar un *alias* del ambiente del usuario, se utiliza el comando *unalias,* en este ejemplo se borrará el *alias pasa*:

```
$ unalias  pasa
```

3.2.7 Comando history

El interprete shell mantiene un archivo que guarda el historial de todos los comandos que se escriben y le permite reescribirlos. El archivo que guarda este historial es mantenido durante toda la sesión. Para listar los comando guardados por este archivo se ejecuta el comando history.

COMANDO history

Sintaxis: Donde n es el número comando a desplegar.

Y [a z] es el rango del número de comando a listar.

$ history [*-n* |*a z*]

- El shell mantiene un archivo de historia de los comandos escritos.
- El comando *history* despliega los últimos 16 comandos.
- Reescribe los comando.

Figura 3.2.7 Sintaxis del comando history

El comando *history* desplegará los últimos 16 comandos que se escribieron. Cada línea es precedida con un número de comando, se puede referir a este número de comando cuando se desee reescribir el comando.

Ejemplo comando history:

- Despliega los últimos 2 comandos escritos.

```
$ history -2
16   cd
17   more .profile
```

- Lista los comandos del 3 al 5.

```
$ history 3 5
3     date
4     pwd
5     ls
```

- Reescribir el comando pwd.

```
$ r  6
pwd
/users/pepe/pruebas/depto1/caso1
```

CAPÍTULO 4. Tips de Administración

4.1 Herramientas para formatos de texto

Una de las características más valiosas del Sistema UNIX es el amplio repertorio de comandos que proporciona. Este apartado examinará un conjunto de comandos particularmente útiles que se conocen como herramientas. Son una colección de órdenes pequeñas, cada una de ellas realiza una función específica, tal como ordenar una lista, buscar una palabra en un archivo o unir dos archivos sobre un campo común. Pueden usarse individualmente o en combinación para llevar a cabo muchas de las tareas habituales.

4.1.1 Búsqueda de patrones dentro de un archivo

Entre las herramientas más útiles en el Sistema UNIX están las encargadas de encontrar las palabras en archivos: *grep, fgrep* y *egrep*. Estas órdenes buscan cadenas que coincidan con un objetivo o patrón especifico, para extraer información de archivos, buscar líneas relacionadas con un elemento en particular y para localizar archivos que contengan una palabra clave.

- **grep** es el comando más usado de los tres. Permite buscar palabras o patrones que contengan comodines y otros elementos de expresiones regulares.
- **fgrep**, busca objetivos múltiples, pero no permite expresiones regulares.
- **egrep** usa un conjunto más rico de expresiones regulares, permite búsquedas con objetivos múltiples y es considerablemente más rápido que *grep*.

grep

La orden *grep* busca en uno o más archivos líneas que contengan un objetivo, como palabras claves o cadenas de texto, como resultado, despliega las líneas completas que cumplan con el objetivo. También permite buscar patrones con expresiones regulares (. * [ ] a-z etc.).

Agumentos:

-v Encontrar líneas que no cumplan con el objetivo.
-i Ignorar distinciones entre mayúsculas y minúsculas.
-l Imprime sólo nombres de archivos.
-n Lista sólo los números de línea en los que se encuentra el objetivo.

Ejemplo comando grep:

- En el siguiente ejemplo se muestran todas las líneas de archivo1 que contienen la palabra estudiante.

```
$ grep estudiante  archivo1
El estudiante trabaja en el ensayo1
Pedro es estudiante de primer año
Lola es estudiante de segundo año
```

- En este ejemplo se imprimen todas las líneas de archivo1 que contienen la frase “es estudiante”, si la cadena tiene espacios en necesario encerrar entre comillas.

```
$ grep “es estudiante”  archivo1
Pedro es estudiante de primer año
Lola es estudiante de segundo año
```

- Se puede dar a *grep* dos o más archivos de búsqueda, este incluirá el nombre del archivo delante de cada línea al desplegar el resultado.

```
$ grep estudiante archivo1 archivo2
archivo1: El estudiante trabaja en el ensayo1
archivo1: Pedro es estudiante de primer año
archivo1: Lola es estudiante de segundo año
archivo2: Manolo estudiante de medicina
```

fgrep

La orden *fgrep* es similar a *grep* pero con tres diferencias principales:

- Se puede utilizar para buscar varios objetivos al mismo tiempo.
- No permite utilizar expresiones regulares.
- Es más rápido que *grep*.

A continuación se muestran un ejemplo del comando fgrep:

Ejemplo comando fgrep:

- En el siguiente ejemplo se muestran todas las líneas de archivo1 que contienen la palabra estudiante y medicina.

```
$ fgrep “estudiante
› medicina” archivo1
El estudiante trabaja en el ensayo1
Lola es estudiante de segundo año
Manolo estudiante de medicina
```

Nótese que cuando se da a *fgrep,* varios patrones de búsqueda debe colocarse entre comillas y cada uno debe estar en línea separada.

egrep

La orden *egrep* contiene a *grep* y *fgrep,* lo que lo hace más potente al combinarlos y facilita la sintaxis. El comando egrep utiliza todas las opciones y sintaxis de grep y fgrep para buscar, modificar, etc. patrones dentro de un archivo.

Ejemplo de egrep:

```
$ egrep “es estudiante|medicina|cocina” *
archivo1: Pedro es estudiante de primer año
archivo1: Lola es estudiante de segundo año
archivo2: Manolo estudiante de medicina
archivo3: Paty estudiante de cocina
```

4.1.2 Trabajo en columnas y campos en archivos

Muchos archivos contienen información organizada en términos de posiciones dentro de una línea. Entre ellos se encuentran los archivos que contienen filas y columnas. Se pueden utilizar las órdenes descritas en esta sección para extraer, modificar o registrar la información de estos archivos estructurados.

- **cut** Permite seleccionar columnas o filas particulares de archivos.
- **paste** Crea nuevas tablas o archivos juntando columnas o campos de archivos existentes.
- **join** Mezcla la información de dos archivos para crear un nuevo archivo con la información de ambos.

cut

Para seleccionar ciertos campos o columnas contenidos dentro de un archivo, *cut* permite extraer esta información. Cuando se utiliza *cut*, hay que indicar cómo identificar los campos (por la posición de los caracteres o mediante el uso de caracteres separadores de campos) y qué campos seleccionar.

Agumentos:

-f Corta el campo indicado de la línea o registro.
-d (Delimitador) Trata a cualquier caracter, como separador de campo.
-c (Columna) Identifica los campos en términos de ls posición de los caracteres dentro de una línea.

Ejemplo de cut:

- Desplegamos el contenido del archivo1

```
$ cut  archivo1
ALUMNOGRADO      MATERIA    PROM.
Pedro        primero     ciencias:mate     8.5
Memo           segundo    español:mate      7.5
Lalo         tercero         ingles:mate       9.0
```

- Muestra sólo la columna 1 y la 4

```
$ cut  -f1,4 archivo1
ALUMNOPROM.
Pedro        8.5
Memo           7.5
Lalo         9.0
```

- Seleccionar la tercer columna y mostrar la primer materia de cada alumno:

```
$ cut –d: -f3 archivo1
MATERIA
ciencias
español
ingles
```

- Quita los primeros 15 caracteres.

```
$ cut –c15  archivo1
RADO    MATERIA        PROM.
rimero       ciencias:mate     8.5
egundo  español:mate      7.5
ercero       ingles:mate       9.0
```

paste

Este comando une archivos línea a línea. Se puede usar para crear nuevas tablas juntando campos o columnas de dos o más archivos.

Argumentos:

-d (Delimitador) Trata a cualquier carácter como separador de campo.

Ejemplo de paste:

- Juntar la información del archivo1 y archivo2 en archivo3.

```
$ cat archivo1
ALUMNOGRADO      MATERIA        PROM.
Pedro        primero     ciencias:mate     8.5
Memo           segundo    español:mate      7.5
Lalo         tercero         ingles:mate       9.0
Dany         primero     ciencias:mate     8.5

$ cat  archivo2
ALUMNO     CICLO   MATRI.
Pedro        2000     15958-0
Memo           2001     14885-1
Lalo         2002     14258-2
Dany         2001     13256-1

$ paste  archivo1 archivo2 > archivo3
$ cat  archivo3
ALUMNO    GRADO   MATERIA  PROM.  ALUMNO   CICLO    MATRI.
Pedro        primero   ciencias:mate   8.5         Pedro       2000 15958-0
Memo           segundo  español:mate    7.5            Memo       2001 14885-1
Lalo         tercero    ingles:mate      9.0         Lalo         2002 14258-2
Dany         primero   ciencias:mate   8.5         Dany        2001 13256-1
```

- Juntar la información del archivo1 y archivo2 en archivo4 separando cada campo con “:”.

```
$ paste  -d: archivo1 archivo2 > archivo4
$ cat  archivo4
ALUMNO:GRADO:MATERIA:PROM.:ALUMNO:CICLO:MATRI.
Pedro:primero:ciencias:mate:8.5:Pedro:2000:15958-0
Memo:segundo:español:mate:7.5:Memo:2001:14885-1
Lalo:tercero:ingles:mate:9.0:Lalo:2002:14258-2
Dany:primero:ciencias:mate:8.5:Dany:2001:13256-1
```

join

La orden *join* une dos archivos a partir de un campo en común. Es similar a *paste,* pero *join* identifica líneas idénticas (campo clave), el campo común aparece sólo una vez en la salida.

Agumentos:

-j Usa otro campo indicado como campo común.

Ejemplo de paste:

- Juntar la información del archivo1 y archivo2 en archivo3.

```
$ cat archivo1
ALUMNOGRADO      MATERIA        PROM.
Pedro        primero     ciencias:mate     8.5
Memo           segundo     español:mate      7.5
Lalo         tercero          ingles:mate       9.0
Dany         primero     ciencias:mate     8.5

$ cat  archivo2
ALUMNO    CICLO    MATRI.
Pedro           2000     15958-0
Memo       2001     14885-1
Lalo         2002     14258-2
Dany           2001     13256-1

$ join archivo1 archivo2 > archivo3
$cat  archivo3
ALUMNOGRADO      MATERIA    PROM.   CICLO    MATRI.
Pedro        primero     ciencias:mate     8.5       2000     15958-0
Memo           segundo     español:mate      7.5       2001     14885-1
Lalo         tercero          ingles:mate       9.0       2002     14258-2
Dany         primero     ciencias:mate     8.5       2001     13256-1
```

4.1.3 Ordenando datos en archivos

La ordenación de información en archivos y el trabajo sobre archivos ordenados son algunas de las tareas más comunes. La información se ordena en un archivo para facilitar la lectura, para procesar la información de entradas o salidas ordenadas. La información se puede ordenar alfabéticamente, numéricamente, por líneas o en función de un campo o columna en particular. Las herramientas más útiles del Sistema UNIX son:

- **sort** Ordena o reordena las líneas de un archivo.
- **uniq** Filtra o elimina las líneas repetidas de los archivos.

sort

Es una herramienta potente de propósito general para ordenar información.

Agumentos:

-d Ordena sólo sobre letras, dígitos y blancos.
-f Ignora la distinción entre mayúsculas y minúsculas.
-m Ordena cadenas como “Jan” o “Feb” por su orden en el calendario.
-n Ordena por valor numérico en orden ascendente.
-r Invierte el orden de sort.
-u (uniq) Elimina las líneas repetidas en el archivo.

Ejemplo de sort:

- En el ejemplo siguiente se ordenará alfabéticamente el contenido del archivo1, primero utilizamos el comando *cat* para ver el contenido del archivo1 y después se ejecuta el comando *sort*:

```
$ cat  archivo1
Pedro
Memo
Lalo
Paty

$ sort archivo1
Lalo
Memo
Paty
Pedro
```

- En el siguiente ejemplo sort ignora las primeras tres columnas y ordena los datos con la información de la columna cuatro:

```
$ cat  archivo1
Pedro     primero      ciencias:mate 8.5
```

```
Memo      segundo    español:mate 7.5
Lalo      tercero    ingles:mate   9.0
Paty      segundo    español:mate 8.0

$ sort +3 archivo1
Memo      segundo    español:mate 7.5
Paty      segundo    español:mate 8.0
Pedro     primero    ciencias:mate 8.5
Lalo      tercero    ingles:mate   9.0
```

uniq

El comando *uniq* elimina las líneas repetidas en un archivo. *Uniq* también dispone de diferentes opciones útiles.

Argumentos:

-c Cuenta el número de veces que se repite una línea.
-d (duplicadas) Muestra solamente las líneas repetidas.
-u (únicas) Imprime las líneas que sólo aparecen una vez.

Ejemplo de uniq:

- En el siguiente ejemplo se borran las líneas repetidas en el archivo1 y se mostrará un conteo de cuantas líneas estaban repetidas.

```
$ cat  archivo1
8.5
7.5
9.0
8.5
9.0

$ uniq –c archivo1
2   8.5
1   7.5
2    9.0
```

4.1.4 Comparación de archivos

Con frecuencia se necesita ver si dos archivos tienen diferente contenido. Las órdenes descritas en esta sección se pueden utilizar para comparar contenidos de archivos.

- **cmp, comm y diff** Informan si dos archivos son el mismo o diferentes y proporciona información de dónde y como difieren. Se distinguen por la cantidad de información que proporcionan.

cmp

Informa el lugar dentro del archivo donde ocurre la primera diferencia.

Ejemplo de cmp:

- En este ejemplo se compararán dos archivo, con el comando *more* se desplegara el contenido de los archivos: archivo1 y archivo2, para verificar la diferencia:

```
$ more  archivo1
Pedro    primero     ciencias:mate     8.5
Memo       segundo    español:mate      7.5
Lalo      tercero          ingles:mate        9.0

$ more  archivo2
Pedro    primero     ciencias:mate     5.5
Memo       segundo    español:mate      7.5
Lalo      tercero          ingles:musica      9.0

$ cmp archivo1 archivo2
archivo1  archivo2  differ: char  67, line 1
```

comm

Sirve para comparar dos archivos ordenados. El comando *comm* imprime su salida en tres columnas: líneas únicas en el primer archivo, líneas únicas en el segundo y líneas de ambos archivos.

Argumentos:

-1 Elimina los informes de las líneas únicas del primer archivo.
-2 Elimina los informes de las líneas únicas del segundo archivo.
-3 Elimina la impresión de las líneas de ambos archivos.

Ejemplo de comm:

- En este ejemplo se comparan el archivo1 y el archivo2, pero antes se desplegará la información de cada archivo para interpretar mejor el resultado del comando *cmm*.

```
$ cat  archivo1
Pedro
Memo
Lalo
Paty

$ cat  archivo2
Pedro
Memo
Lalo
Dany

$ comm archivo1 archivo2
        Pedro
    Memo                <-------------  Se encuentran en los dos archivos
        Lalo
Paty                    <-------------  Solo se encuentra en el archivo1
    Dany                <-------------  Solo se encuentra en el archivo2
```

diff

Compara dos archivos línea a línea e imprime las diferencias. Además para cada línea de texto diferente en los dos archivos, *diff* muestra las diferencias.

Salida:

- #c# Cambio entre el # de línea del primer archivo < y el # de línea del segundo archivo >
- #a# Indica que el # de línea del primer archivo no existe en el segundo y se necesita añadir en el # de línea del segundo archivo.
- #d# Indica líneas que se encuentran en un archivo pero no en otro.

Ejemplo de diff:

- El siguiente ejemplo compara las líneas del archivo1 y el archivo2 y muestra en que difieren estos archivos.

```
$ cat  archivo1
Pedro    primero     ciencias:mate 8.5
Memo       segundo    español:mate 7.5
Lalo      tercero          ingles:mate    9.0

$ cat  archivo2
Pedro    primero     ciencias:mate 5.5
Memo       segundo    español:mate 7.5

$ diff  archivo1 archivo2
1c1
< Pedro primero     ciencias:mate 8.5
---
>
Pedro    primero     ciencias:mate 5.5
3d3
> Lalo       tercero          ingles:mate    9.0
```

4.1.5 Cambio de información en archivos

A veces se necesita modificar el contenido de un archivo añadiendo, eliminando o cambiando su información, pero sin abrir el archivo, es decir, sin editarlo. El Sistema UNIX proporciona diferentes herramientas para la edición no interactiva. En esta sección describiremos dos de ellas:

- **tr** Cambia o traduce cualquier caractere procedente de la entrada de acuerdo a reglas que se especifican.
- **sed** Incluye casi todas las funciones de edición que se encuentran en los editores *vi* y *ed*, pero en forma no interactiva.

tr

tr sólo lee la entrada estándar (datos en línea ingresados por el usuario), de manera que hay que utilizar la redirección para proporcionar la entrada del archivo “>”. La mejor forma de explicar lo que hace *tr* es mediante un ejemplo simple:

Argumentos:

-s Elimina los caracteres repetidos en la cadena.
-c Caracteres no se deben sustituir.
-d Borra en la salida caracteres del conjunto de entrada.

Ejemplo de tr:

- El archivo1 usa dos puntos (:) para separar campos, los cambiaremos por tabuladores.

```
$cat  archivo1
ALUMNO:GRADO:MATERIA:PROM.
Pedro:primero:ciencias:mate:8.5
Memo:segundo:español:mate:7.5
Lalo:tercero:ingles:mate   9.0:2002
Dany:primero:ciencias:mate:8.5

$tr : ‘<TAB>’ < archivo1
$cat  archivo1
ALUMNO   GRADO   MATERIA PROM.
Pedro   primero   ciencias:mate     8.5
Memo      segundo  español:mate     7.5
Lalo     tercero    ingles:mate        9.0
Dany    primero   ciencias:mate     8.5
```

- Ahora se cambiarán las letras minúsculas a mayúsculas del archivo1:

```
$tr '[a-z]'  '[A-Z]'  < archivo1
$cat  archivo1
ALUMNO    GRADO      MATERIA          PROM.
PEDRO     PRIMERO    CIENCIAS:MATE    8.5
MEMO      SEGUNDO    ESPAÑOL:MATE     7.5
LALO      TERCERO    INGLES:MATE      9.0
DANY    PRIMERO    CIENCIAS:MATE    8.5
```

sed

sed es una herramienta potente para filtrar archivos de texto. Se puede utilizar para realizar cambios globales que involucran caracteres individuales, palabras o patrones y para hacer cambios específicos del contexto incluyendo eliminación o adición de texto. Las órdenes de *sed* constan generalmente de una dirección y una orden, a continuación se definen estas dos funciones:

- *dirección*: Número de líneas a utilizar y patrones de expresiones regulares.
- *orden*: borrado, adición o cambio de información de la línea de texto.

Argumentos:

-n (no copiar) Sólo pasan a la salida las líneas que se especifican.

Ejemplo de sed:

- En este ejemplo se añadirán dos líneas al final del archivo1. Se utiliza el carácter "\" para seguir ingresando los datos en el siguiente renglón.

```
$ cat  archivo1
ALUMNO   GRADO   MATERIA     PROM.
Pedro     primero       ciencias:mate     8.5
Memo       segundo       español:mate      7.5
```

```
$ sed '$a Lalo   tercero   ingles:mate      9.0 \
Dany   primero   ciencias:mate   8.5 '   >  archivo1

$ cat  archivo1
ALUMNO   GRADO   MATERIA     PROM.
Pedro    primero     ciencias:mate     8.5
Memo       segundo    español:mate     7.5
Lalo      tercero      ingles:mate       9.0
Dany     primero     ciencias:mate     8.5
```

- Sustituir 8.5 por 9.0 en el archivo1.

```
$ sed 's/8.5/9.0/g' archivo1
$ cat  archivo1
ALUMNO   GRADO   MATERIA     PROM.
Pedro      primero    ciencias:mate     9.0
Memo       segundo       español:mate     7.5
Lalo       tercero         ingles:mate       9.0
Dany      primero        ciencias:mate     9.0
```

- Borra las líneas 2 y 3 del archivo1

```
$ sed '2,3 d' archivo1
$ cat  archivo1
ALUMNO   GRADO   MATERIA     PROM.
Lalo          tercero    ingles:mate          9.0
Dany         primero   ciencias:mate    9.0
```

En la siguiente tabla se muestra un resumen básico de las funciones del comando *sed*:

Orden	Función
A	Añade
I	Inserta
C	Cambia.
D	Borra.
S	Sustituye
P	Imprime
Q	Salir
R	Lee
W	Escribe

Tabla 4.1.5 Funciones del comando sed

4.2 Utilidades

El Sistema UNIX proporciona un sin número de utilidades que pueden ser utilizadas para realizar un amplio rango de tareas y para resolver una gran variedad de problemas. A continuación se detallaran algunas de estas utilidades.

4.2.1 Cálculos matemáticos

Se pueden utilizar varias herramientas proporcionadas por el Sistema UNIX para realizar cálculos aritméticos básicos o complejos. Entre estas herramientas se encuentran *dc*, *bc* y *factor*. En este capítulo veremos la opción *bc* que es el método más familiar para utilizar.

- **bc** Es un programa potente, flexible y de alta precisión para realizar cálculos aritméticos. Es una calculadora y un pequeño lenguaje para escribir programas numéricos.

bc

Proporciona todas las operaciones aritméticas estándares y proporciona cualquier grado de precisión. Automáticamente visualiza la salida de cada línea.

Ejemplo bc:

- Se realizará la operación “((8+4)*3)/2” y se obtendrá el resultado en la última línea:

```
   $ bc
((8+4)*3)/2
16
```

4.2.2 Visualizar fecha y hora

El sistema UNIX proporciona dos utilidades para obtener información sobre la fecha y hora.

- **cal** Imprime un calendario de cualquier mes o año.
- **date** Imprime la fecha y hora actuales en diferentes formatos.

cal

Si no se da ningún argumento *cal* imprime el calendario completo del año actual. Se introduce en primer lugar el mes y luego el año.

Ejemplo de cal:

- Con el comando *cal* se obtendrá el calendario del mes de noviembre del año 1996.

```
$cal 11 1996
    November 1996
S   M   Tu  W   Th  F   S
                1   2   3   4
 5   6   7   8   9  10  11
12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30
```

date

Es usado para poner o cambiar la hora del sistema, cambiar el formato de la hora y fecha del sistema. Algunos de las especificaciones más comunes se muestran en la siguiente tabla:

Utilidad	Descriptor	Ejemplo
Año	y Y	96 1996
Mes	M b B	11 Nov November
Día	a A	Sat Saturday
Día del mes	d	18
Hora	I H T	05(01 al 12) 17(00 al 23) 5:23:15
Minuto	M	23
Segundo	S	15
A.M./P.M.	P	pm
Fecha Num.	D	11/18/72

Tabla 4.2.2. Especificaciones del comando date

Ejemplo de date:

- Mostrar la hora y fecha del sistema.

```
$ date
Sat Nov18 17:19:33 EST 2006
```

- Desplegar un nuevo formato de la hora y fecha del sistema.

```
$ date ″+Hoy es %A, %B %d, %Y″
Hoy es Wednesday, November 8, 2006
```

4.2.3 La orden script

La orden *script* copia todo lo que se realiza y visualiza en la pantalla en un archivo, incluyendo la orden de entrada, la petición de orden del sistema y la salida. Puede ser muy útil para revisar como se resolvió un problema o para llevar bitácoras. Para utilizarlo, se teclea simplemente el comando *script* seguido del nombre del archivo en el que se almacenará la trascripción. Para finalizar la ejecución del comando se teclea CTRL-D.

Ejemplo de script:

- Guardar todo lo que se realiza en la sesión en un archivo llamado "bitacora1"

```
$ script bitacora1
script started.  file is bitacora1
```

[*Todo lo que se teclee incluyendo comandos y salidas, errores y controles como CTRL-M y RETURN, etc*]

```
script  completes.  file is bitacora1
```

4.2.4 Control de Procesos

En el Sistema UNIX se asigna un número a cada proceso ya sean propios del sistema o ejecutados por usuarios o programas, así como bases de datos, etc. El número asignado de identificación es único, conocido como PID. Conocer este número de procesamiento resulta de mucha utilidad para el manejo de los siguientes comandos, que en este apartado se verán en forma general:

- **ps** Reporta el estado del proceso.
- **nohup** Impide que un proceso termine al salir de la sesión.
- **Kill** Termina un proceso.

ps

La opción *ps* reporta el estado del proceso, el propietario, la terminal en la cual se esta ejecutando el proceso, muestra el número asignado al proceso PID, etc.

Ejemplo de ps:

- Lista todos los procesos que se están corriendo en el sistema.

```
$ps -fea
UID   PID    PPID   C  STIME     TTY     TIME  COMMAND
root  1332   1       0  18:31:12       ttyp2  0:15  cat
dany  1384   0       0   Jan 27      ttyp2  0:03  vi
```

nohup

El comando *nohup* hace que los comandos o script´s (programas) sigan ejecutándose aún después de suspender o al salir de la sesión (log out). El comando *nohup* puede direccionar su salida a un archivo, si no es así, el comando nohup lo direccionará automáticamente a un archivo llamado nohup.out.

Ejemplo de nohup:

- Ejecutar el script prog1 utilizando el comando *nohup* para poder salir del sistema y guardar el resultado del script en el archivo1:

```
$nohup prog1 > archivo1 &
[1] 972
```

kill

El comando *kill* se utiliza para terminar cualquier comando incluyendo comandos nohup, *kill* no puede cancelar comandos de otros usuarios a menos que sea root el que ejecute el *kill*. Existen varios argumentos para utilizar el *kill,* pero la opción más cercana a una cancelación segura es *kill* -9.

Ejemplo de kill:

- Terminar la ejecución del script prog1

```
$ nohup prog1 > archivo1
[1] 972
$kill -9 972
```

- Terminar el editor vi, primero se obtiene el PID para poder eliminar este proceso:

```
$ ps -fea
UID   PID    PPID   C  STIME     TTY     TIME  COMMAND
root  1332   1       0  18:31:12       ttyp2  0:15  cat
dany 1384   0       0   Jan 27      ttyp2  0:03  vi
$kill -9  1384
```

4.3 Servicios de Red

Cuando se utilizan más de una computadora y es necesario que se comuniquen entre sí, dichos equipos deberán formar parte de una red LAN, WAN, MAN, etc. De la cuales no se hablará en este tutorial.

Sin embargo es necesario conocer en forma general algunos comandos de red que proporciona el Sistema UNIX:

- **hostname** Reporta el nombre de la computadora en red.
- **telnet** Ingresa a otra computadora remotamente.
- **ftp** Copia archivos de o hacia una computadora remota.
- **rlogin** Ingresa a otra computadora remotamente.
- **ping** Determina si se puede contactar a otro equipo dentro de la red.

4.3.1 Comando hostname

La computadora o servidor tiene un nombre interno que es el *hostname* y es el que identifica a la computadora en la red.

Ejemplo de hostname:

- Para obtener el nombre de un servidor se utiliza el comando hostname:

```
$ hostname
soporte10
```

- El archivo /etc/hosts contiene el nombre de las computadoras o servidores registradas para comunicarse como se ve en el siguiente ejemplo:

```
$ more /etc/hosts

192.1.2.1   soporte10
192.1.2.2     respaldo10
192.10.2.1       monterrey1
```

4.3.2 Comando telnet

Este comando es un login (entrada) remoto a otro servidor, es necesario conocer un usuario del servidor remoto y la contraseña para poder accesar. Una vez entrando al servidor remoto se puede trabajar normalmente como si se estuviera localmente en ella. Para salir del comando *telnet* es con *exit*.

Ejemplo de telnet:

- Conectarse del servidor soporte10 al servidor mty10

```
$ hostname
soporte10
 $ telnet mty10
 Trying…
 Coneccted to mty10
 Escape character is ´^]´

 login: user1
 Passwd:
```

- Se verifica el ingreso al servidor mty10 y después se sale de él:

```
$ hostname
 mty10
$ exit              (salir del telnet)
```

4.3.3 Comando ftp

Para copiar un archivo de un servidor o computadora a otra remota se utiliza *ftp* (file transfer protocolo). Para utilizar este comando es necesario conocer como en el *telnet* un usuario y el password. El comando f*tp* cuenta con un gran número de comandos para realizar el trabajo, algunos ejemplo son:

get	Este copia un archivo de la máquina remota a la local.
put	Copia un archivo de la máquina local a la remota.
ls	Lista los archivos de la máquina remota.
?	Lista todos los comandos del ftp.
quit	Desconecta y termina el ftp.

Ejemplo de ftp:

- Enviar el archivo1 del servidor soporte10 a mty10

```
 $ hostname
 soporte10
$ ftp mty10
Connected to mty10
220 mty10 FTP server (Version 2.9: $DATE 88/12/21 10: 19 &)
ready.
Name: user1
Password:
331 Password requiered for user1
230 User user1 logged in.
ftp> send archivo1 /tmp
200 PORT  command okay.
150 Opening data connection for /tmp/archivo1 (192.10.2.1,1041)
226 Transfer complete.
396 bytes sent in 0.19 seconds (20.53 Kbytes/sec)
ftp>quit
200 Goodbye.
$
```

4.3.4 Comando rlogin

El comando *rlogin* realiza funciones similares al *telnet*. Pero con este comando se ingresará automáticamente al sistema remoto. Esto es posible si el administrador del sistema remoto tiene un archivo llamado /etc/hosts.equiv configurado. Este tampoco pedirá password si se configura en el directorio *home* el archivo .rhosts. Esto lo hace muy fácil y rápido de usar.

Ejemplo de rlogin:

- Conectarse al servidor soporte10 al servidor mty10, verificar el ingreso al servidor y salir:

```
$ hostname
soporte10
$ rlogin mty10
$
 $ hostname
 mty10
 $ exit
```

4.3.5 Comando ping

Este comando sirve para determinar si uno hosts (IP) esta activa dentro de la red para ellos envía llamadas determinadas hacia el equipo remoto utilizando,

solicitando una respuesta al host remoto a dicha petición. Si el equipo remoto esta activo la respuesta será contestada de forma inmediata asegurando que hay conectividad entre ambos equipos.

Ejemplo de ping

- Verificar que el host cuya dirección IP 10.10.10.1 esta activa dentro de la red

```
# ping 10.10.10.1

10.10.10.1 is alive
```

CONCLUSIÓN:

Este trabajo pretende ser un punto de partida para usuarios principiantes que busquen formarse como administradores del sistema operativo UNIX Solaris ya que la mayoría de los manuales son altamente especializados, complicando con ello el aprendizaje de un usuario inexperto.

Este tutorial abarcó los comandos más relevantes e importantes para empezar a ser un administrador del sistema operativo UNIX Solaris de manera sencilla y fácil de aprender, ya que como administrador es necesario contar con información del manejo del sistema de manera ágil, esto es sin tener que consultar grades libros y se vuelva una inmensidad de información.

Un administrador al consultar este tutorial adquiriro en muy poco tiempo el manejo del sistema operativo UNIX, ya que su contenido es la experiencia de varios años de trabajo diario, por ello se plasmó en este, información relevante y ligera de manejar, guiando al administrador aprendiz de la manera más clara posible, al uso del sistema operativo UNIX

ANEXO A. Glosario de términos

Este glosario contiene definiciones de términos importantes utilizados en el presente tutorial, además se aportan algunas otras que no están incluidas en este trabajo ya que son usadas en el ámbito informático.

a.out

Nombre asignado al archivo de salida creado durante la compilación, ensamblaje o carga de un archivo fuente de lenguaje C.

acotación

Uso de caracteres especiales para instruir al shell de que el contenido dentro de los caracteres debe ser tratado como una sola cadena.

administración de sistemas

Mantenimiento de los archivos, usuarios y procesos en un sistema. Mientras los usuarios puedan efectuar sencillas tareas de administrativas, la administración compleja y rutinaria se efectúa por el *administrador del sistema.*

administrador del sistema

La persona que mantiene un sistema, incluyendo la preparación de los entornos de usuario, el mantenimiento de los recursos para los usuarios del sistema y la sintonía del sistema para mejorar su rendimiento.

admintool

Admintool es el ambiente gráfico para el manejo de comandos, por ejemplo para añadir un usuario se puede realizar gráficamente (manejo de ventanas y menú)

alias

Un nombre alternativo creado por el usuario para una orden o cadena. También la orden para crear o visualizar alias. Los alias se utilizan con frecuencia para cadenas de órdenes largas o complejas o direcciones de correo largas.

alias de orden

Un alias o alternativa asignada a una orden. Se utiliza con frecuencia para reducir la escritura necesaria para construir una *línea de orden*.

aplicación

Programa que realiza una o más funciones específicas, como contabilidad o procesamiento de texto.

archivar

Almacenar datos en un medio destinado a almacenamiento a largo plazo (cintas, C.D., discos, etc.).

archivo

Una secuencia de bytes dentro del sistema de archivos referenciados por su *nombre de archivo*. Los archivos pueden ser *ordinarios* (ASCII o binarios) o especiales (entrada y salida) . Los archivos informáticos facilitan una manera de organizar los recursos usados para almacenar permanentemente información dentro de una computadora.

archivo binario

Generalmente, un archivo almacenado en código máquina.

archivo de cabecera

Un archivo que contiene definiciones de código fuente de macros y variables. El nombre del archivo de cabecera puede ser incluido al comienzo de un programa desarrollado por un usuario. Los archivos, también llamados "*archivos de inclusión*", tienen nombres que generalmente acaban con .h, indicando que son archivos de cabecera (heder).

archivo de entorno

Un archivo que contiene parámetros que definen el entorno de un usuario bajo el *Shell Korn*. Junto con los valores de las variables especificadas en los archivos *.profile* de los usuarios, el archivo de entorno puede restringir o ampliar diferentes capacidades de usuarios en el mismo sistema.

archivo ejecutable

Un archivo de texto o archivo binario que tiene permisos establecidos para permitir su ejecución escribiendo simplemente su nombre.

archivo especial

Un tipo de archivo que contiene información acerca de un dispositivo tal como un disco o un terminal de usuario. Los archivos especiales son utilizados por el sistema para operaciones de entrada y salida.

archivo especial de bloque

Un archivo UNIX que trabaja como interfaz entre el Sistema Operativo UNIX y bloques de dispositivo de entrada y salida, como discos y cintas que soportan un sistema de archivos.

archivo oculto

Un archivo cuyo nombre comienza con un punto (.) se denomina oculto ya que los nombres de los archivos que comienzan con un punto no son incluidos en la salida de ls (a menos que se suministre un argumento, tal como –a, que diga a ls que incluya los archivos ocultos). Un ejemplo es el archivo *.profile.*

archivo ordinario

Un archivo que contiene datos, ordenes shell o programas. Los archivos ordinarios son creados y mantenidos por los usuarios.

archivo .rc (archivo run command)

Un archivo guión que contiene parámetros u ordenes que se ejecutan al comienzo de una cierta orden con el fin de preparar un entorno bajo el cual ejecutar esa orden. Ejemplos de archivos *.rc* son el archivo *.mailrc*, utilizado por la orden mailx, el archivo *.exrc* utilizado por ex y vi, y el archivo *.newsrc,* utilizado por readnews.

argumento (de una orden).

Una palabra por ejemplo un nombre de archivo, que forma parte de una línea de orden. El Shell interpreta los argumentos de las órdenes para ver si la sintaxis es correcta antes de ejecutar la orden.

argumento cero

El nombre de la orden en una línea de orden. Este es el valor de la variable $0.

ARPANET

Red creada por DARPA que conecta aproximadamente 150 lugares en universidades y corporaciones que realizan investigación para el gobierno de los Estados Unidos. La ARPANET utiliza el protocolo TCP/IP. Forma parte de Internet.

arranque inicial (boot)

El proceso de inicialización que carga el núcleo, inicializa la memoria, inicia la ejecución de los procesos del sistema y prepara el entorno del usuario.

array

Estructura de datos que trata a los elementos contiguos como una serie de patrones repetidos. Cada elemento de un array se referencia mediante un índice, que proporciona la situación del elemento en relación a otros elementos.

array asociativo

Un array en el que conjuntos de datos se asocian arbitrariamente por campos o pares de cadenas **awk** y **perl** utiliza arrays asociativos para permitir la flexibilidad en el procesamiento.

ASCII

(American Standard Code for Informarion Interchange). Un código de caracteres estándar utilizado en muchos sistemas informáticos. El ASCII tradicional utiliza únicamente siete bits de los ocho posibles para presentar datos, pero el ASCII extendido utiliza los ocho bits para definir caracteres adicionales.

asignación de memoria

Método por el cual el núcleo determina que paginas de memoria se asignan a que procesos.

biblioteca

Un archivo de funciones comúnmente utilizado en programación. Los directorios de bibliotecas generalmente tienen nombres que comienzan con lib, y suele ser compatibles por muchos usuarios.

bloque

Grupo de datos tratado como una unidad durante las operaciones de entrada/salida (E/S). Los dispositivos de disco y cinta se llaman dispositivos de bloques, indicando que leen y escriben bloques de datos cada vez.

búfer

Una posición de almacenamiento temporal utilizada para contener datos antes de transferirlos desde un área hasta otra. Los buffers se utilizan principalmente para aumentar la eficiencia durante los procesos de entrada/salida. El núcleo utiliza buffers para mover bloques de datos entre procesos y dispositivos tales como discos y terminales.

bug

Un error de programación que produce resultados impredecibles, como la detención de todos los procesos en ejecución. Los bugs pueden ser trazados y corregidos utilizando un *depurados.*

buzón

Un archivo en un directorio designado por el usuario o el sistema como lugar para contener el correo de llegada.

caballo de Troya

Un programa que se enmascara como otro programa y efectuá alguna funcion desconocida para la persona que lo ejecuta. Un virus puede ser esparcido mediante un programa caballo de Troya.

cliente/servidor

Arquitectura que divide la ejecución de procesos en una red entre una computadora o computadoras que solicitan servicios llamada “cliente” y estas proporcionan los servicios, llamada “servidor”.

código fuente

El código fuente es un conjunto de líneas que conforman un bloque de texto, escrito según las reglas sintácticas de algún lenguaje de programación destinado a ser legible por el programador.

código de terminación

Salida de código de un proceso o programa que indica el estado del proceso; por ejemplo, si se ha completado con éxito o no. Los códigos de terminación se utilizan para realizar ejecución condicional de programas subsiguientes.

comando

Una orden (a veces llamada comando) es una instrucción o mandato que el usuario proporciona al sistema, desde la línea de órdenes o una llamada a programa, el cual generalmente está contenido en un archivo ejecutable.

compilador

Programa que convierte un código fuente a un código binario.

compresión

Un método de reducción de tamaño de archivos para almacenamiento. Los contenidos del archivo son con frecuencia comprimidos utilizando un algoritmo para reemplazar el código ASCII con palabras código de longitud variable.

configuración

Personalización del sistema operativo para que realice las tareas requeridas.

CPU

La unidad central de proceso (CPU), o algunas veces simplemente llamadas procesador, es el componente en un computador digital que interpreta las instrucciones y procesa los datos contenidos en los programas de computador.

consola

La terminal principal, utilizado por el administrador del sistema para monitorear procesos y peticiones de usuario o para efectuar funciones de administración del sistema.

contraseña

Cadena de caracteres que debe suministrarse cuando el usuario ingresa al sistema. Las contraseñas proporcionan seguridad frente a accesos no autorizados.

controlador de dispositivo

Programa que permite al sistema UNIX controlar la comunicación entre una computadora y sus dispositivos periféricos.

copia de seguridad (Backup)

Guardar una copia de un archivo, directorio o sistema de archivos completos. Las copias de seguridad son importantes en caso de fallo del sistema, además de ser un modo de restaurar el sistema.

cron

Una utilidad empleada para planificar procesos para ejecución rutinaria

crontab

Una tabla planificada de procesos configurable por el usuario.

cuenta de usuario

Una cuenta de usuario en un sistema UNIX almacena los datos, configuraciones y ambiente de trabajo de un usuario en particular.

cuota

Espacio en disco previamente determinado por el administrador para el uso exclusivo de uno o más usuarios de un grupo

depurador

Un paquete que muestra el camino lógico, valores de registros y variables durante la ejecución de un proceso o programa para determinar donde se produce un fallo. El Sistema UNIX tiene un depurador llamado mdb que puede ayudar a encontrar que instrucciones se estaban ejecutando durante un volcado de memoria.

delimitador

Caracteres utilizados para separar campos o cadenas dentro de un archivo o líneas de comando, el shell utiliza el espacio en blanco como delimitador implícito de palabras.

demonio

Un demonio es un proceso, puede ser un proceso planificado regularmente. Este tiene la característica de que siempre esta corriendo en la memoria del sistema.

DISPLAY

Variable de ambiente, cuyo valor por default apunta al Xwindow local del sistema.

editor de texto

Es un programa que permite escribir y modificar archivos compuestos únicamente por texto con o sin formato, conocidos comúnmente como archivos de texto.

emulador

Es una aplicación que permite ejecutar programas de una arquitectura diferente sobre la cual se están ejecutando.

encryptar

Forma de codificar información.

estructura de árbol

Estructura que se asemeja a un árbol en posición invertida. La estructura de un sistema de archivos dentro de un sistema UNIX maneja una estructura de árbol la cual consta de un directorio raíz y varios subdirectorios.

exportar

Compartir información de un usuario hacia un grupo de usuarios dentro del sistema.

expresión regular

Una expresión utilizada en correspondencia con patrones en archivos. Las expresiones regulares constan de letras y números, además de caracteres especiales que tienen funciones específicas en la búsqueda, llamadas metacaracteres.

filosofía UNIX

La filosofía de que lo pequeño es bello. Las utilidades deberían de ser diseñadas para una sola tarea y de modo que puedan se conectadas. El diseño del Sistema UNIX se basa en la idea de que un sistema informático complejo y poderoso puede seguir siendo sencillo, general y extensible, con el fin de beneficiar tanto a usuarios como creadores. Todo en UNIX es un archivo y si no lo es, entonces es un proceso.

ftp

File Transfer Protocol. Protocoló de red que permite el envió y la recepción de archivos.

host

Equipo de cómputo dentro de una red.

hostname

Nombre de una computadora en una red.

inetd

Demonio encargado de escuchar y atender las peticiones de los servicios de red.

init

Init es el encargado de inicializar todos los procesos del sistema.

initdefault

El valor de initdefault establece el nivel de ejecución por defecto de entrada al sistema.

inode

Todos los archivos en UNIX tienen un inode que mantienen información referente al mismo, tal como situación, derechos de acceso, tamaño o tipo de archivo, etc.

job

Orden interna de bash que muestra los trabajos pendientes que tengamos en segundo o primer plano.

login

Programa encargado de la validación de un usuario al entrar al sistema.

loopback

Sistema de trabajo en red en modo local. Con este sistema podemos trabajar en red con nuestro propio ordenador, su utilidad radica en probar programas de seguridad, leer las noticias o el correo de los servidores instalados en nuestro ordenador o simplemente poder ejecutar Xwindow.

lpd
Demonio encargado de asistir a las peticiones de impresión por parte del sistema.

man
Comando para desplegar los manuales en línea del sistema.

modulos
Porciones de código que se añaden en tiempo de ejecución al kernel (núcleo) del sistema operativo UNIX Solaris para el manejo de dispositivos o añadir funciones al núcleo.

monousuario
Sistema informático que sólo admite a un usuario acceder a los recursos del sistema.

montar
Poner un sistema de archivos en disposición de ser usado por el usuario.

motif
Librería de funciones para el desarrollo de aplicaciones gráficas.

multitarea
Capacidad de un sistema para ejecutar varias procesos al mismo tiempo.

multiusuario
Capacidad de algunos sistemas que ponen disposición sus recursos a varios usuarios a la vez.

núcleo (Kernel)
Parte principal de un sistema operativo, encargado del manejo de los dispositivos, la gestión de la memoria, del acceso a disco y en general de casi todas las operaciones del sistema que permanecen invisibles para nosotros.

PATH
Variable del entorno, cuyo valor contiene los directorios donde el sistema buscara cuando intente encontrar un comando o ejecutable.

PC
Término para definir a una computadora personal (en inglés Personal Computer)

permisos
Todos los archivos en UNIX tienen definido un conjunto de permisos que permiten establecer los derechos de lectura, escritura o ejecución para el propietario del archivo, el grupo al que pertenece y el resto de los usuarios.

PID
Número que identifica a un proceso en el sistema, este número es único para cada proceso.

plataforma
Plataforma se refiere a la arquitectura que se maneja en un sistema.

portabilidad
La portabilidad de un software se refiere a que tan dependiente es una aplicación de la plataforma para la cual fue programada.

prompt
Representación del inicio de sesión, mediante un caracter. Los caracteres más comunes para la representación de sesión son los siguientes: > , $, # .

programa
Un programa informático (software) es la unión de una secuencia de instrucciones que una computadora puede interpretar y ejecutar.

ruta
La ruta es la forma general de localización única de un archivo dentro de un sistema de archivos.

sesión
Sesión es el marco espacio-temporal en el que se desarrolla una actividad concreta, en la cual se participa como actor de la misma.

shell
El shell es el intérprete de comandos usado para interactuar con el núcleo de un sistema operativo.

sistema operativo
Un sistema operativo (SO) es un conjunto de programas destinados a permitir la comunicación del usuario con la computadora y gestionar sus recursos de manera eficiente. Comienza a trabajar cuando se enciende el ordenador, y gestiona el hardware de la máquina desde los niveles más básicos.

superusuario
Normalmente se refiere a la cuenta del administrador del sistema, cuyo nombre convencional en los sistemas UNIX es “root”.

terminal
Un dispositivo utilizado para visualizar la entrada y salida de un sistema conectado. Las terminales pueden ser asíncronas o sincronías; el Sistema UNIX utiliza terminales asíncronas para su visualización.

variables de ambiente
Las variables definen la configuración dentro de la sesión de un usuario. Las cuales se pueden definir automática o manual.

user-name o login

Autenticación (proceso por el cual el usuario se identifica de forma unívoca y en muchos casos sin la posibilidad de repudio).

ANEXO B. Ejercicios

A continuación se verán algunos ejercicios que muestran la funcionalidad y facilidad del manejo de este tutorial.

1.- Cambiar el password de una cuenta en particular.

Respuesta:

```
$ passwd
Changing password :
Old password .
New password :
Re_enter new password :
$
```

2.- Ejecute el comando *date* con las opciones apropiadas de tal forma que la salida este en formato mm-dd-yy.

Respuesta:

```
$ date +%m-%d-%y
```

3.- ¿Cuál es el nombre de su directorio **HOME**?

Respuesta:

```
$ echo $HOME
  /users/estudiante1
```

4.- Desde el directorio HOME, cambiarse al directorio archivo1. Usando una trayectoria relativa, se cambiará al directorio estudiante1. Nuevamente usando una trayectoria relativa, se cambiará al directorio uno.archivo. Finalmente, se regrese al directorio HOME. ¿Qué comandos es utilizado?. Como saber a si se llegó a cada uno de los destinos?

Respuesta:

```
$ cd
$ cd tree/dog.breeds/archivo1
$ cd ../../estudiante1
$ cd ../../uno.archivo
$ cd
```

Para verificar cada destino:

```
$ pwd
```

5.- Crear en el directorio HOME uno archivo llamado archivo1. Realizar en este directorio, un directorio de trabajo actual. ¿Qué comandos se uso? ¿Cuál es la trayectoria completa de este directorio?

Respuesta:

```
$ cd
$ mkdir archivo 1
$ cd archivo 1
$ pwd
/users /estudiante1/archivo
```

6.- En el directorio HOME. Copiar el archivo archivo1 a archivo.prueba. Listar el contenido de ambos archivos para verificar que su contenido es el mismo.

Respuesta:

```
$ cp archivo1 archivo.prueba
   $ cat  archivo1 archivo.prueba
   Hola mundo
   Hola mundo
```

7.- Si el archivo (names) se modifica, afectara esto al archivo names.cp? Modificar el archivo names, copiando el archivo funfile al archivo names. ¿Qué le pasó al archivo names? Y al archivo names.cp?

Respuesta:

```
$ cp funfile names
$ more Hola mundo.cp
```

El archivo names, contiene ahora el mismo contenido de funfile, mientras que names.cp aún contiene lo que estaba en names.

8.- Usar la opción interactiva de *rm* para remover archivo1 y archivo2

Respuesta:

```
$ rm  -i archivo1 archovo2
archivo1? y
archivo2? y
$
```

9.- Copiar el archivo archivo.orig como archivo estudiante.

Respuesta:

```
$ cp archivo.orig estudiante
```

10.- Crear un directorio llamado fruta bajo en el directorio HOME. Con un comando, mover los archivos siguientes, los cuales están también bajo el directorio HOME al directorio fruta:

Lime
Grape
Orange

Respuesta:

```
$ cd
   $ mkdir fruta
   $ mv lime grape orange fruta
```

11.- Mover el directorio fruta desde el directorio HOME al directorio tree.

Respuesta:

```
$ cd
$ mv fruta tree
```

Se creó un directorio fruta desde el directorio HOME al directorio tree.

12.- Enviar el archivo funfile a la impresora en línea. Anotar el request ID que es desplegado en la terminal.

Respuesta:

```
  $ lp funfile
request id is rw.58 (1 file)
```

13.- Verificar que las solicitudes de impresión están encoladas.

Respuesta.

```
$ lpstat
rw -58    ralph  3967     Thu   Jul  04 12:57:25      1992
rw-59     ralph  1331     Thu   Jul  02 13:01:19      1992
```

14.- Usar el comando *cancel* para remover las solicitudes de la cola de la impresora de línea. Y confirmar que fueron canceladas.

Respuesta:

```
$ cancel  rw -58   rw -59
   request ¨rw-58¨ canceled
   request ¨rw-59¨ canceled
   $ lpstat
   $
```

15.- Modifique los permisos de archivo.1 de tal forma que sean: -w-------. Se puede desplegar el contenido de archivo.1?

Respuesta:

```
$ chmod a-rwx,u-w archivo.1
$ cat archivo.1
Can´t open file
```

16.- Modificar los permisos de archivo.1 de tal forma que sean: rw-------. Después desplegar el contenido de archivo.1 ¿Se puede desplegar el contenido de el archivo.1?

Respuesta:

$chmos u=rw archivo.1

No se puede desplegar el contenido de archivo.1

17.- Como se modifican los permisos de archivo.1, de tal forma otro usuario pueda leer el archivo?

Respuesta:

$ chmode g+w mod.1

El archivo permite acceso de lectura a todos los miembros de un grupo .

18.- Hacer una copia del archivo.1 llamado archivo.2. Remover los permisos de escritura del archivo2.puede para poder borrar este archivo ¿Cómo se puede proteger de ser borrado?

Respuesta:

```
$ cp archivo.1   archivo.2
$ chmod –w archivo2
$ rm archivo.2
```

19.- Modificar los permisos de archivo5.dir de tal forma que sean –wx-------. ¿Se pudo desplegar el contenido archivo5.dir? ¿Se puede desplegar el contenido del archivo5.1 bajo el archivo 5.dir?

Respuesta:

```
$ chmode u+wx archivo5.dir
$ ls archivo5.dir
  archivo5.dir unreadable
$ cat archivo5.dir/archivo 5.1
 Este es el contenido de archivo5.
$cd archivo5.dir
$ pwd
 /users/users3/archivo5.1
$ ls
. unreadable
```

19.- Pueden otros usuarios copiar al directorio HOME? ¿Cómo se despliegan los permisos del directorio HOME?

Respuesta:

```
$ cd
$ ls –l .
drwxr-xr 3 estudiante 2 class      1024 Jul   24 13:13
```

BIBLIOGRAFÍA

Referencia de Libros

- The UNIX System.
  Stephen Bourne
  Addison, Wesley Publishing
  ISBN:O-201-13791-7
  Nivel de la Audiencia: Introductorio o intermedio.

- The UNIX Operating System
  Kaare Christian
  John Wiley and Sons Publishing
  ISBN: O-471-89052-9
  Nivel de la Audiencia Introducción o intermedio.

- Unix System handbook for programmers & Administrators.
  Robert B. Bennett.
  McGraw Hill.

Referencia de URL´s

- http://docs.sun.com/app/docs/doc/817-0708/6mgg6t791?q=solaris+9+guide+administration&a=expand

- http://es.wikipedia.org/wiki/Unix

- http://club.telepolis.com/jagar1/Unix/Curso.htm